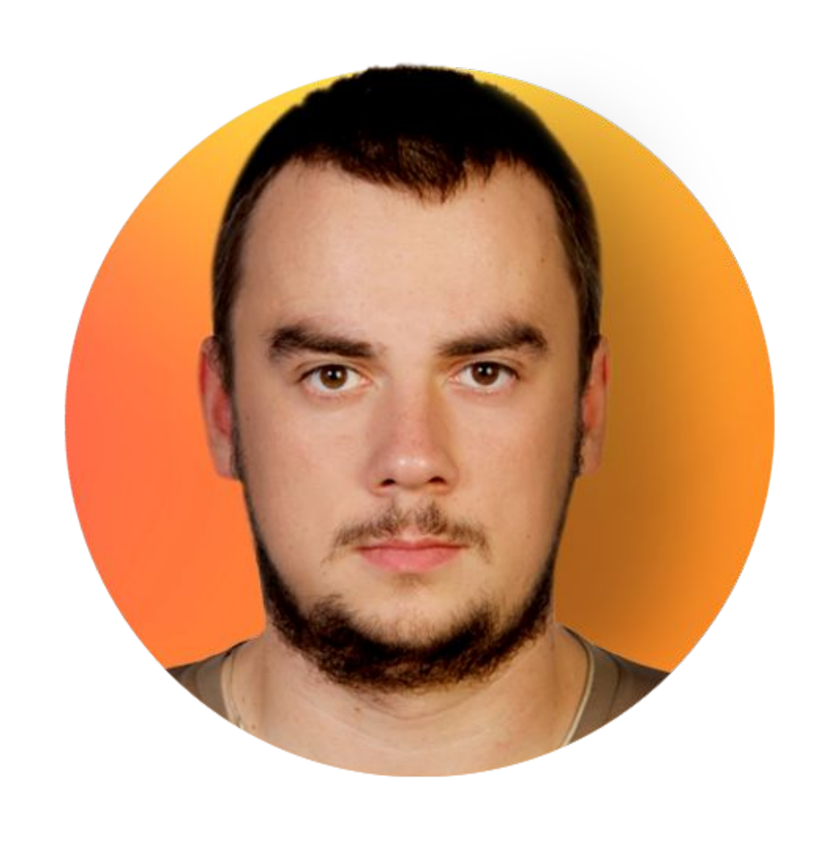
Повышаем производительность файлового I/O в JVM на Linux

Дмитрий Бундин





О спикере

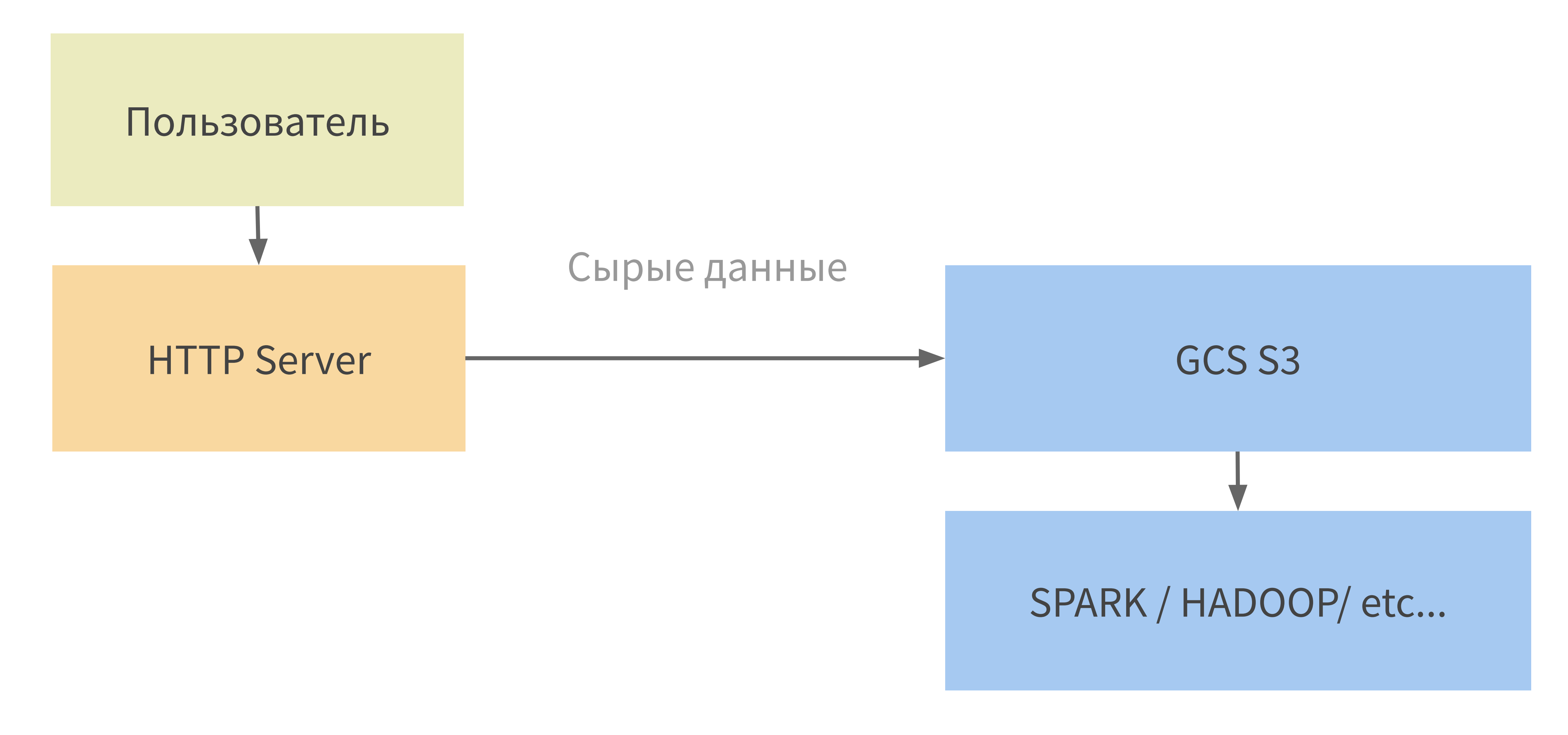


Дмитрий Бундин

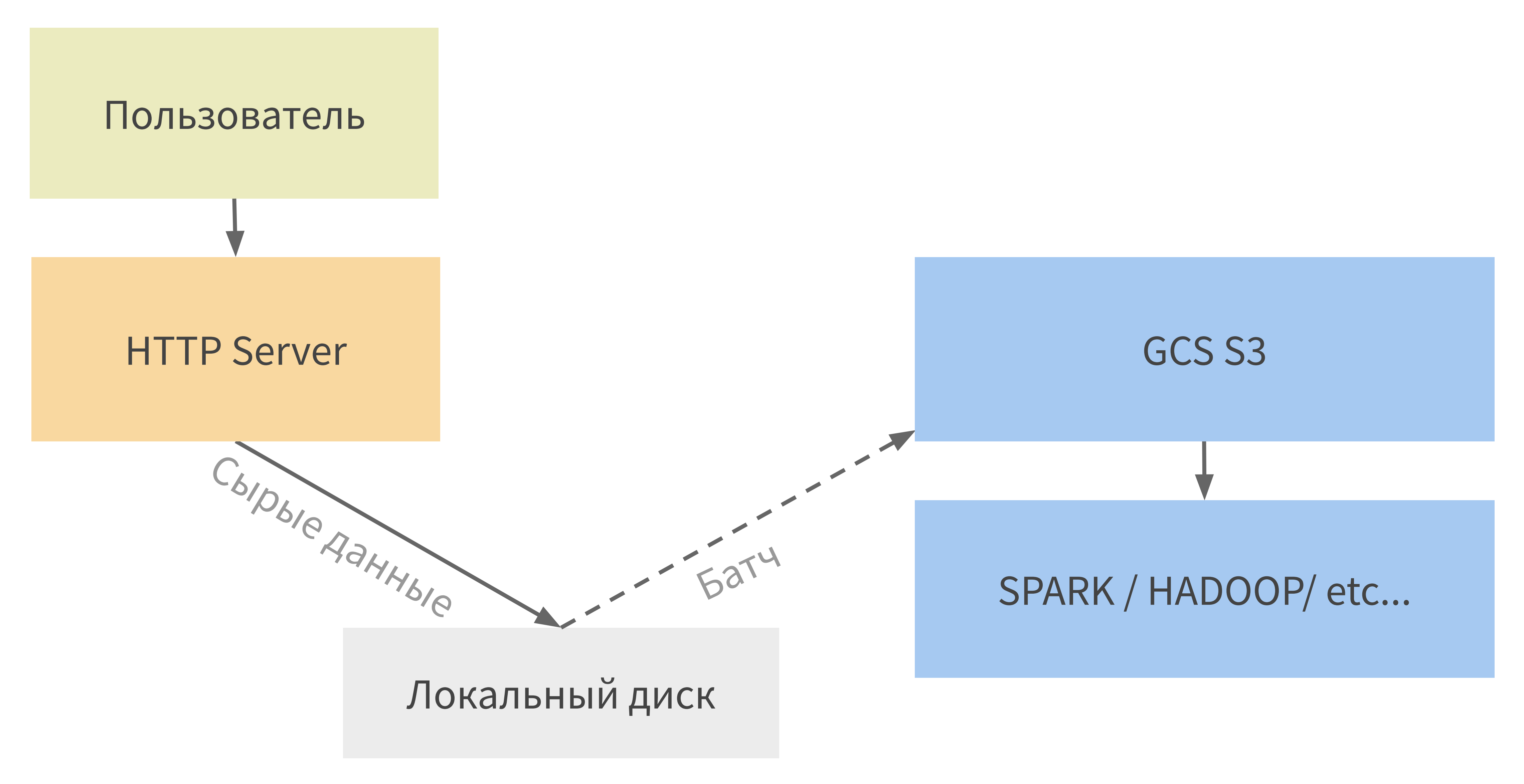
Senior Big Data developer

Дмитрий в ИТ с 2014 года. В последние несколько лет активно занимается разработкой I/O интенсивных приложений на Java, C/Linux и связанными с ними вопросами производительности. До этого разрабатывал банковский софт и платформу обработки данных на Spark в сфере рекламы. Имеет опыт работы со Scala, функциональным программированием и typelevel-стеком. В настоящее время является старшим Big Data-разработчиком в Grid Dynamics.

Проект в сфере рекламы



Проект в сфере рекламы



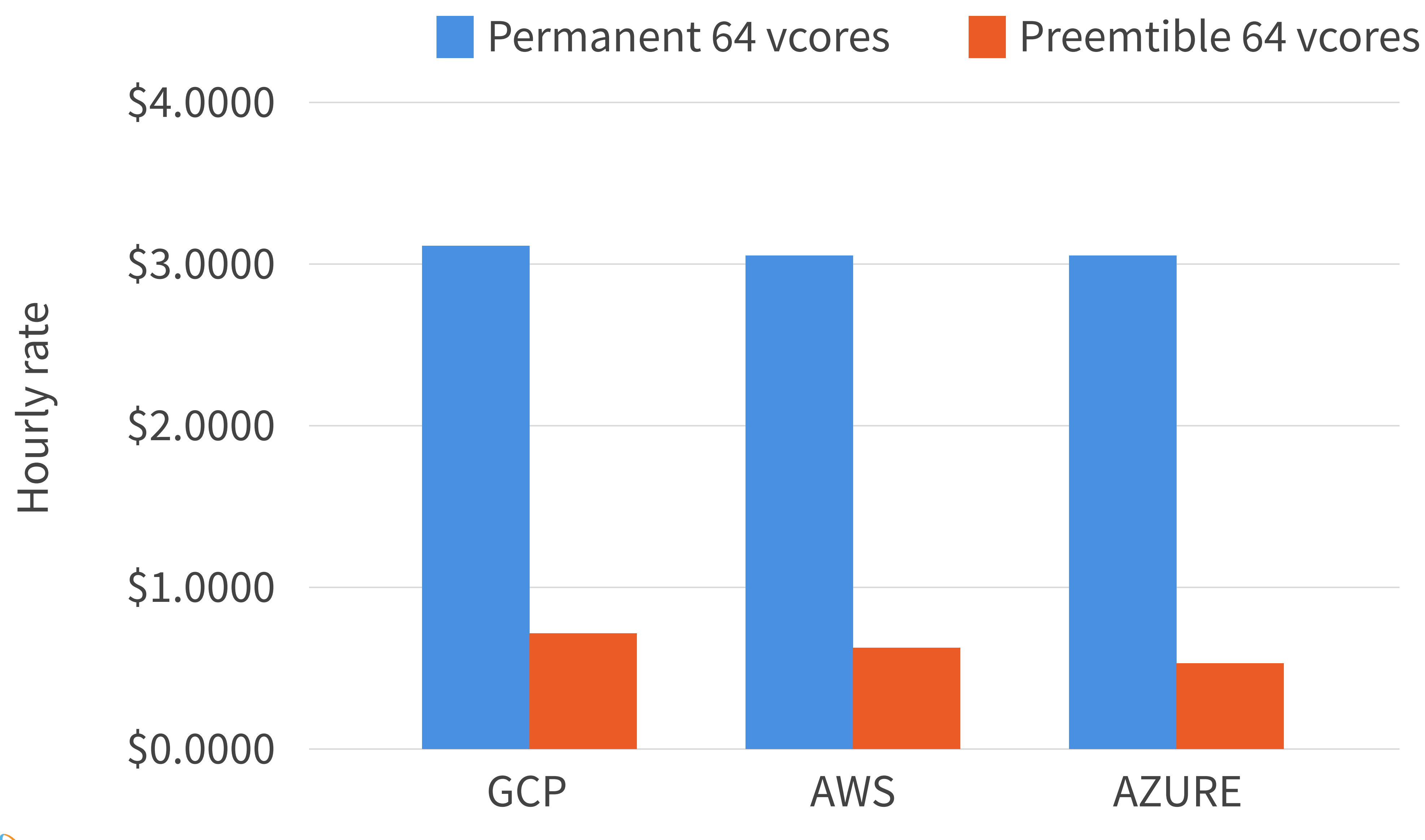
Проект в сфере рекламы: выключение инстанса

HTTP Server → Локальный диск → Система доставки → Внешние хранилища

Permanent instance

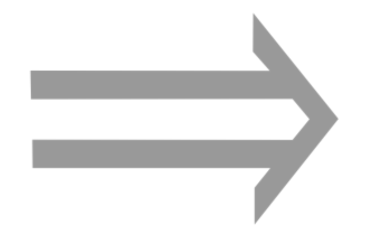


Permanent vs Preemptible: стоимость

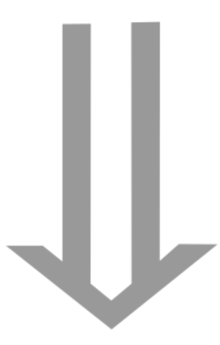


Снижение затрат на инфраструктуру

Permanent Instances

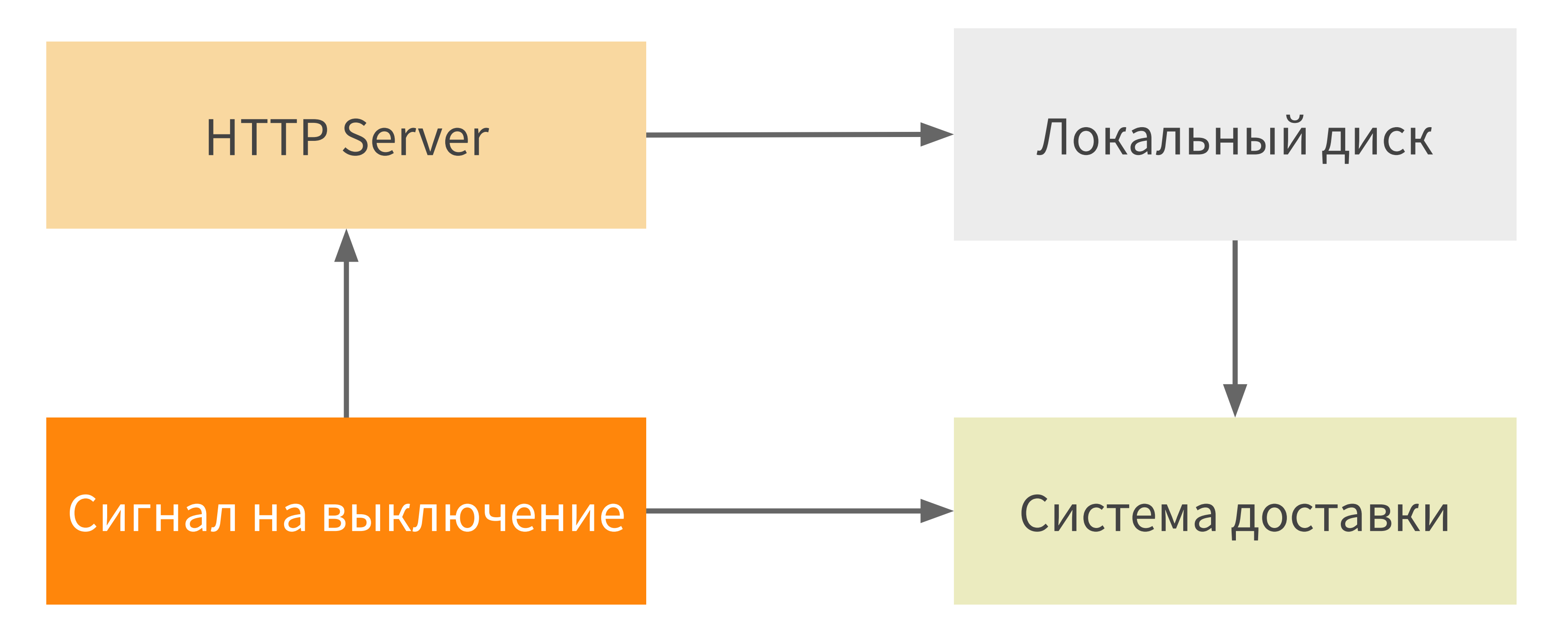


Высокая стоимость

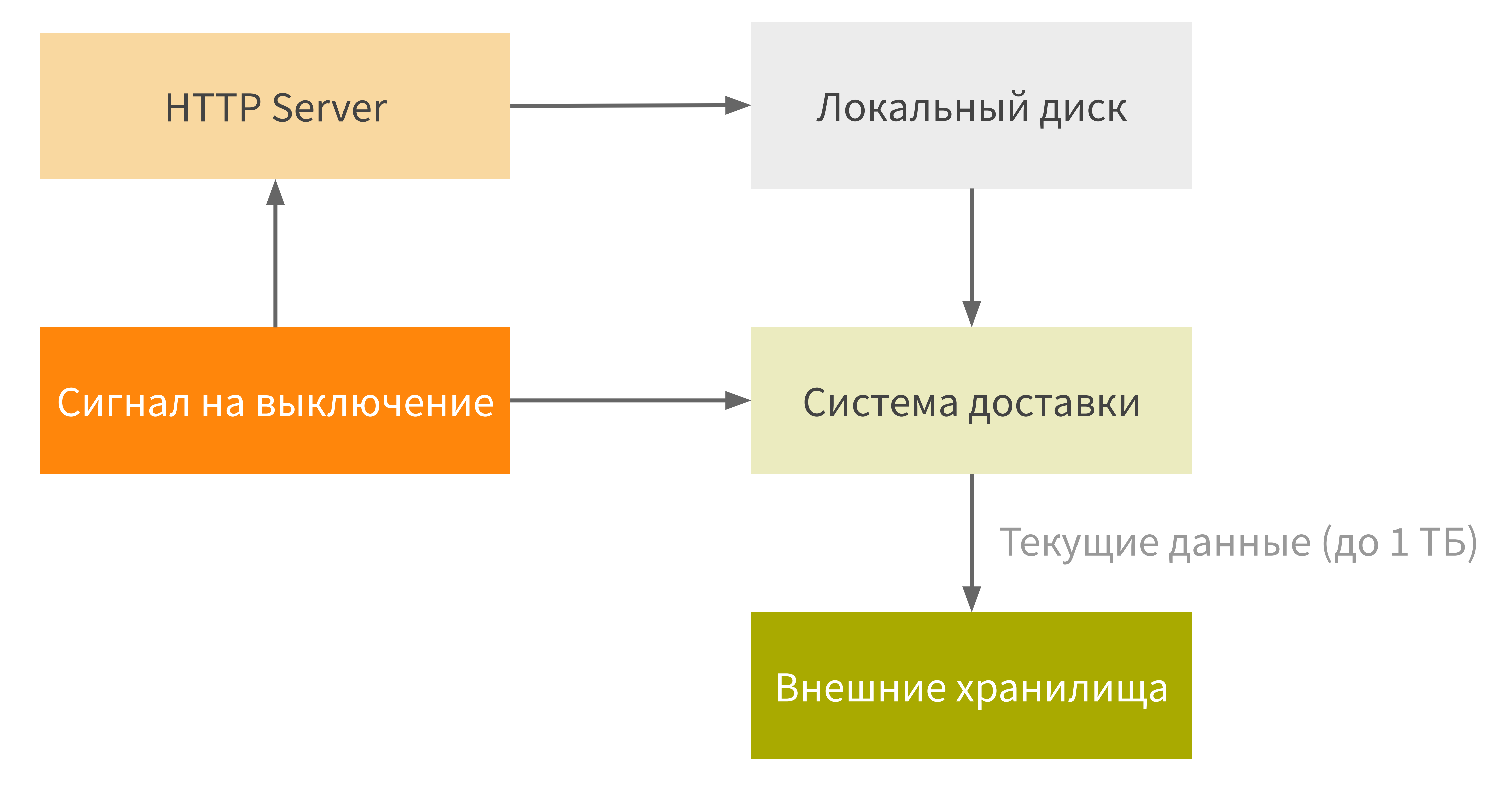


Spot Instances

Проект в сфере рекламы: выключение инстанса



Проект в сфере рекламы: выключение инстанса



Проект в сфере рекламы: выключение инстанса

HTTP Server

Локальный диск

Сигнал на выключение

Система доставки

Потеря данных

Внешние хранилища

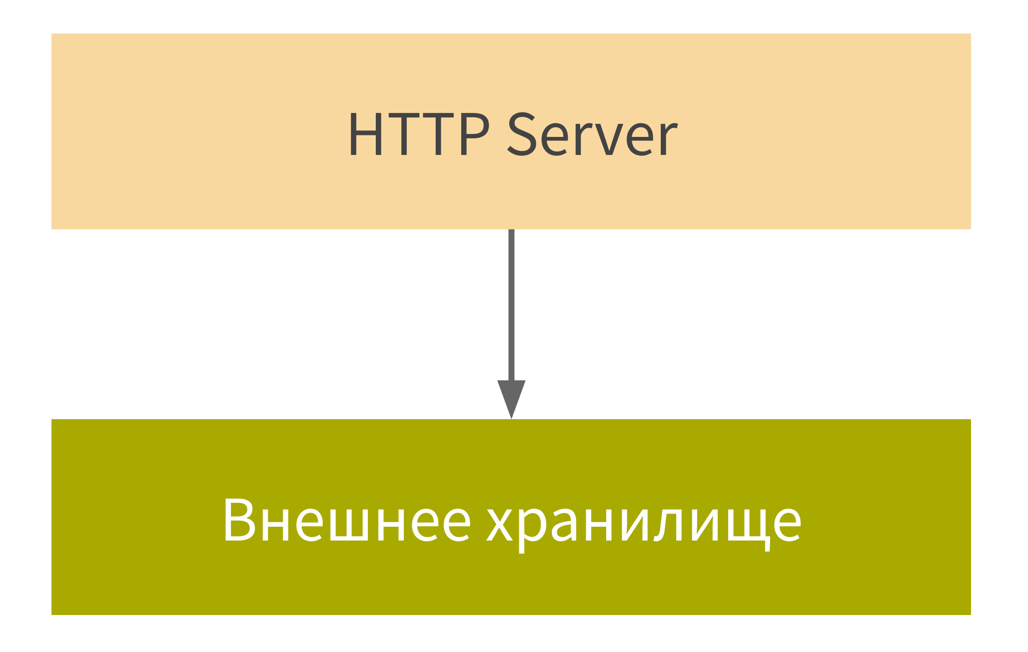
Задачи проекта



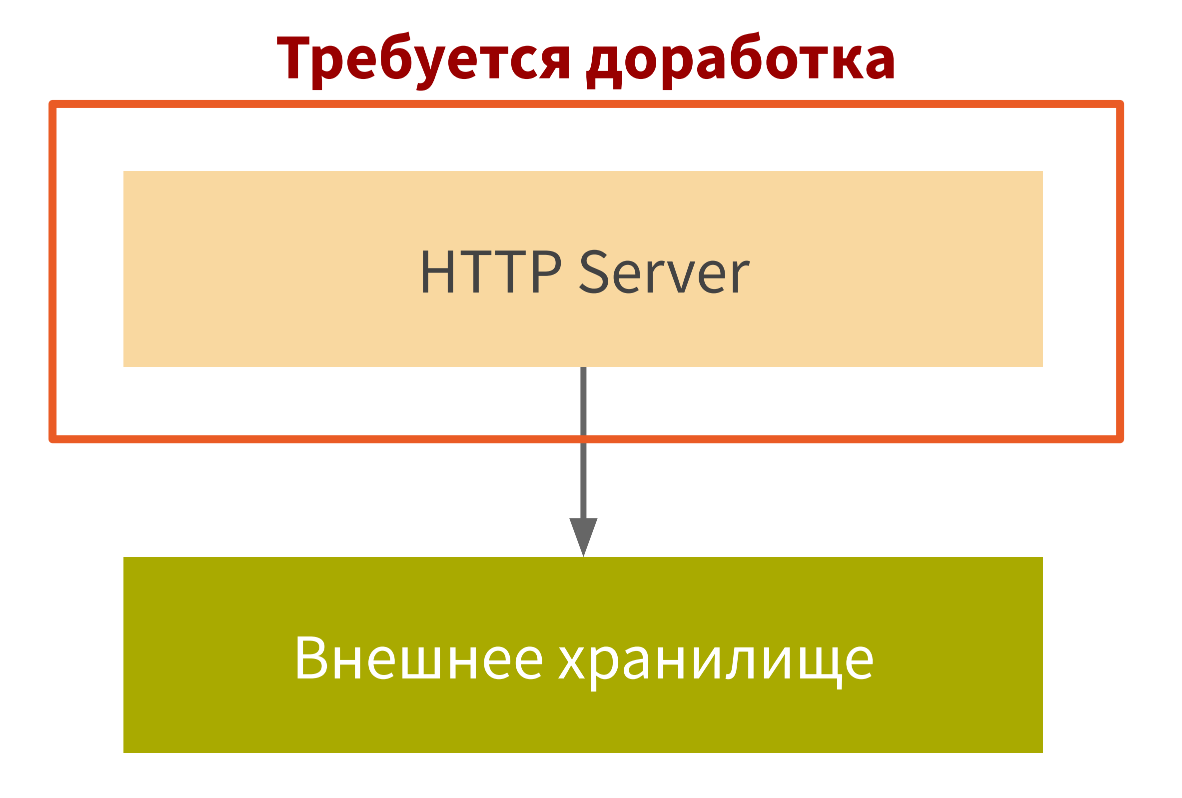


Уменьшение вероятности потери данных

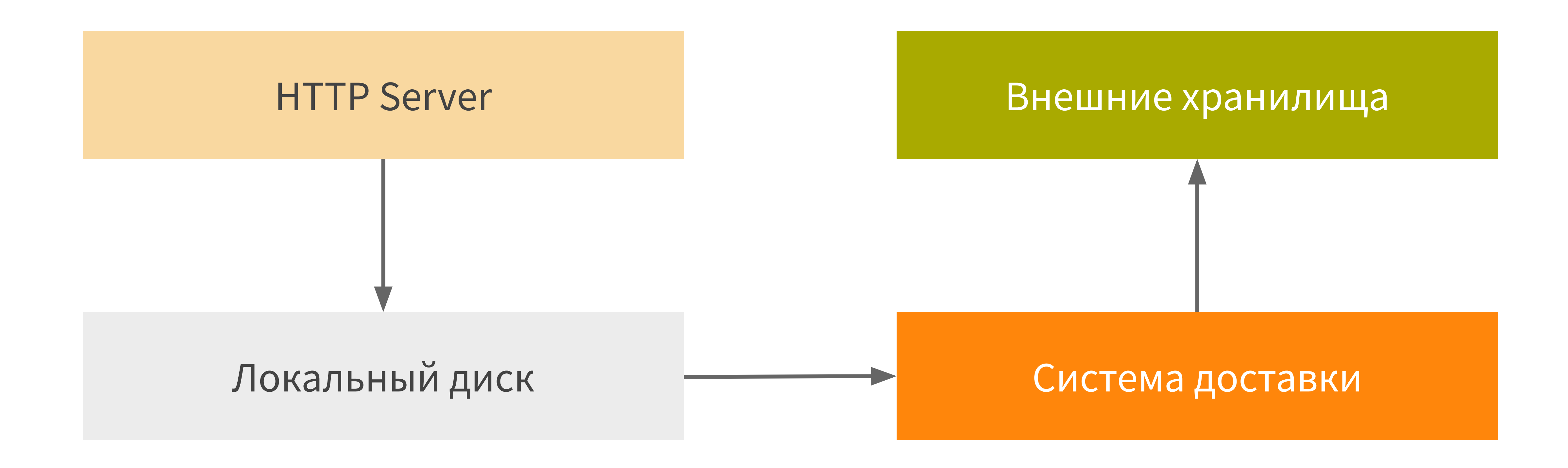
Стриминг данных напрямую



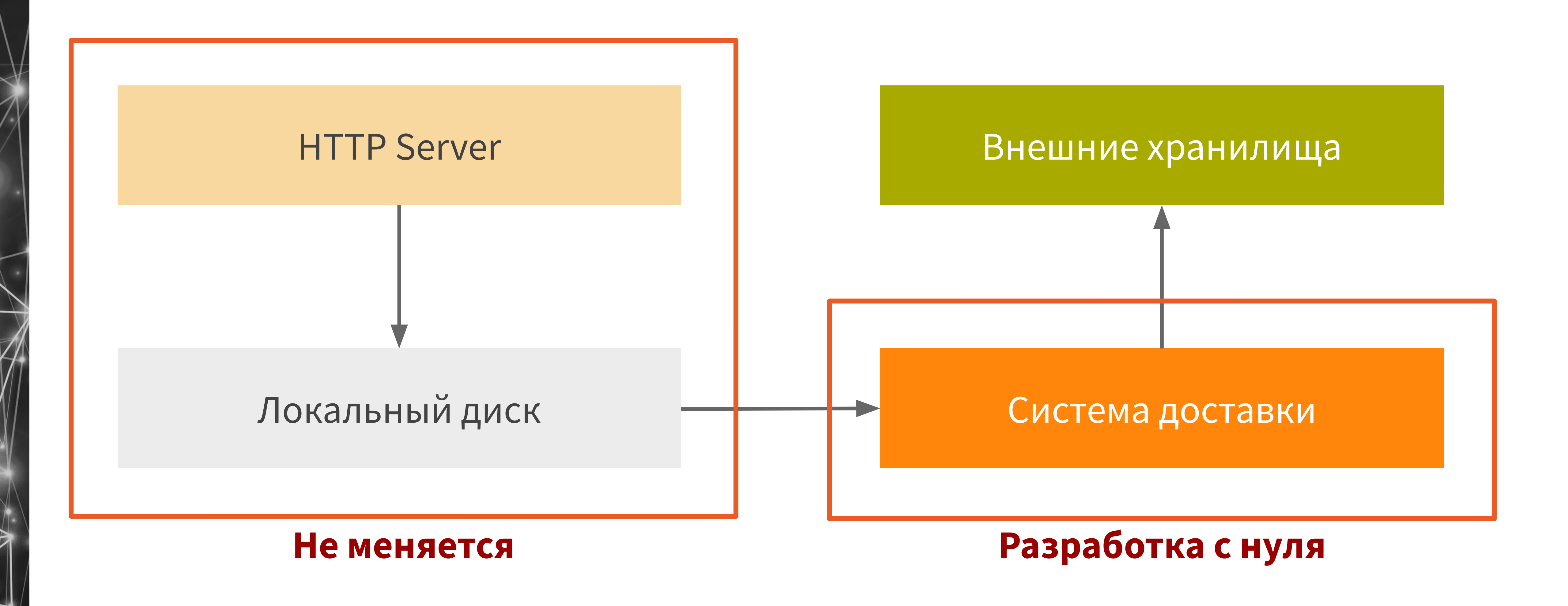
Стриминг данных напрямую



Стриминг данных с локальных дисков



Стриминг данных с локальных дисков



#2 Стриминг данных: сравнение

Стриминг данных напрямую

- Допиливание HTTP Server

- Вероятность потери сохраняется

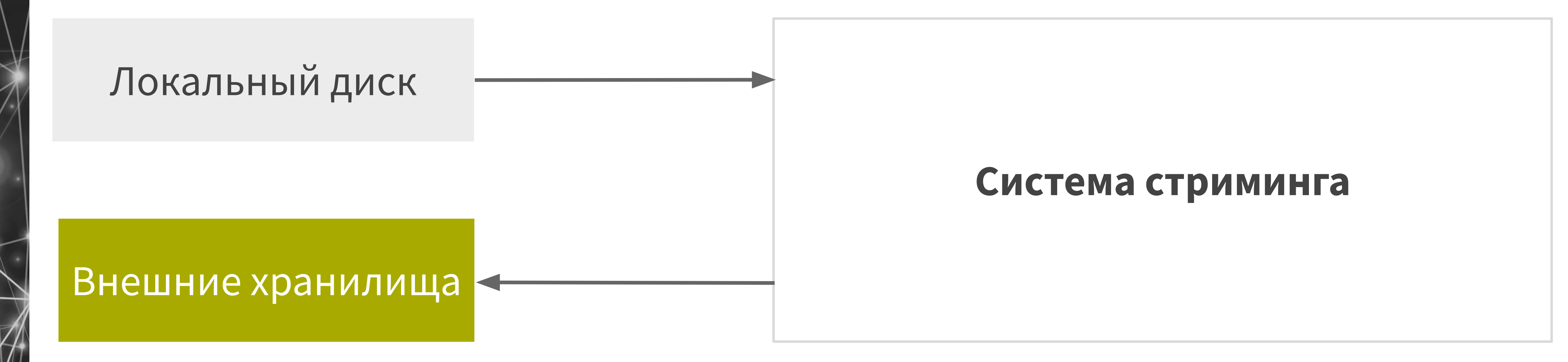
Стриминг данных с дисков

+ Гибкость

~~- Допиливание HTTP Server~~

~~- Вероятность потери сохраняется~~

Схема системы стриминга





Оптимизация доставки данных

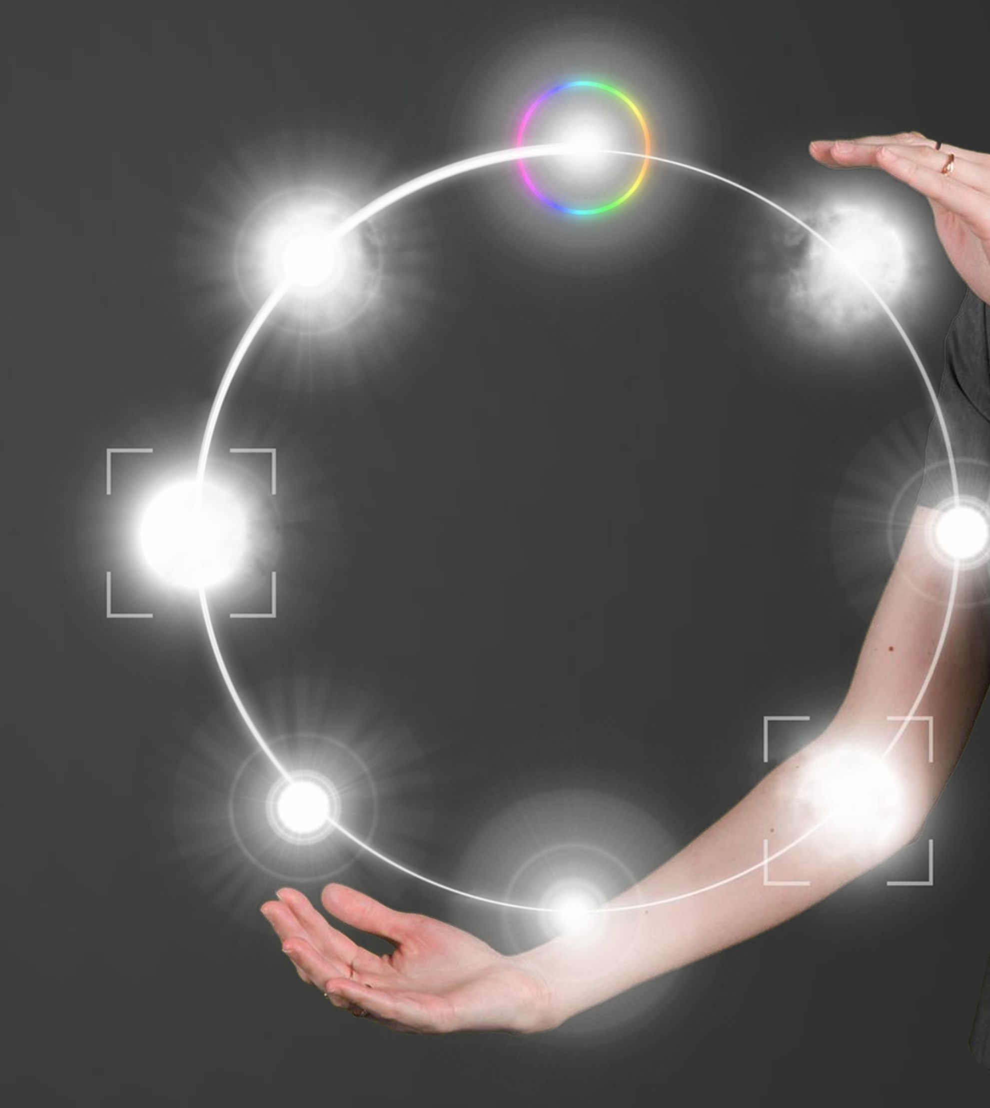


Схема системы стриминга



Схема системы стриминга

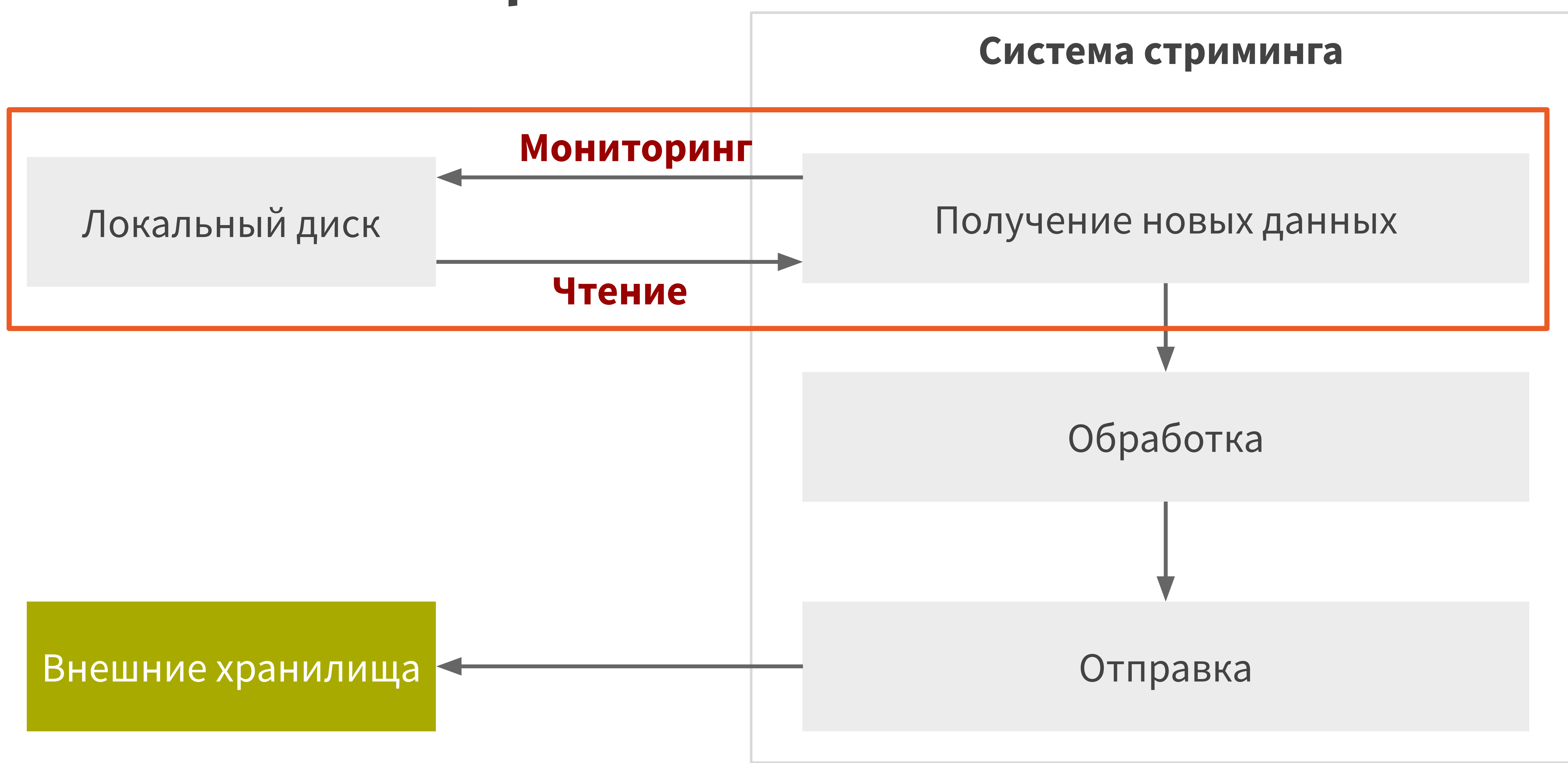
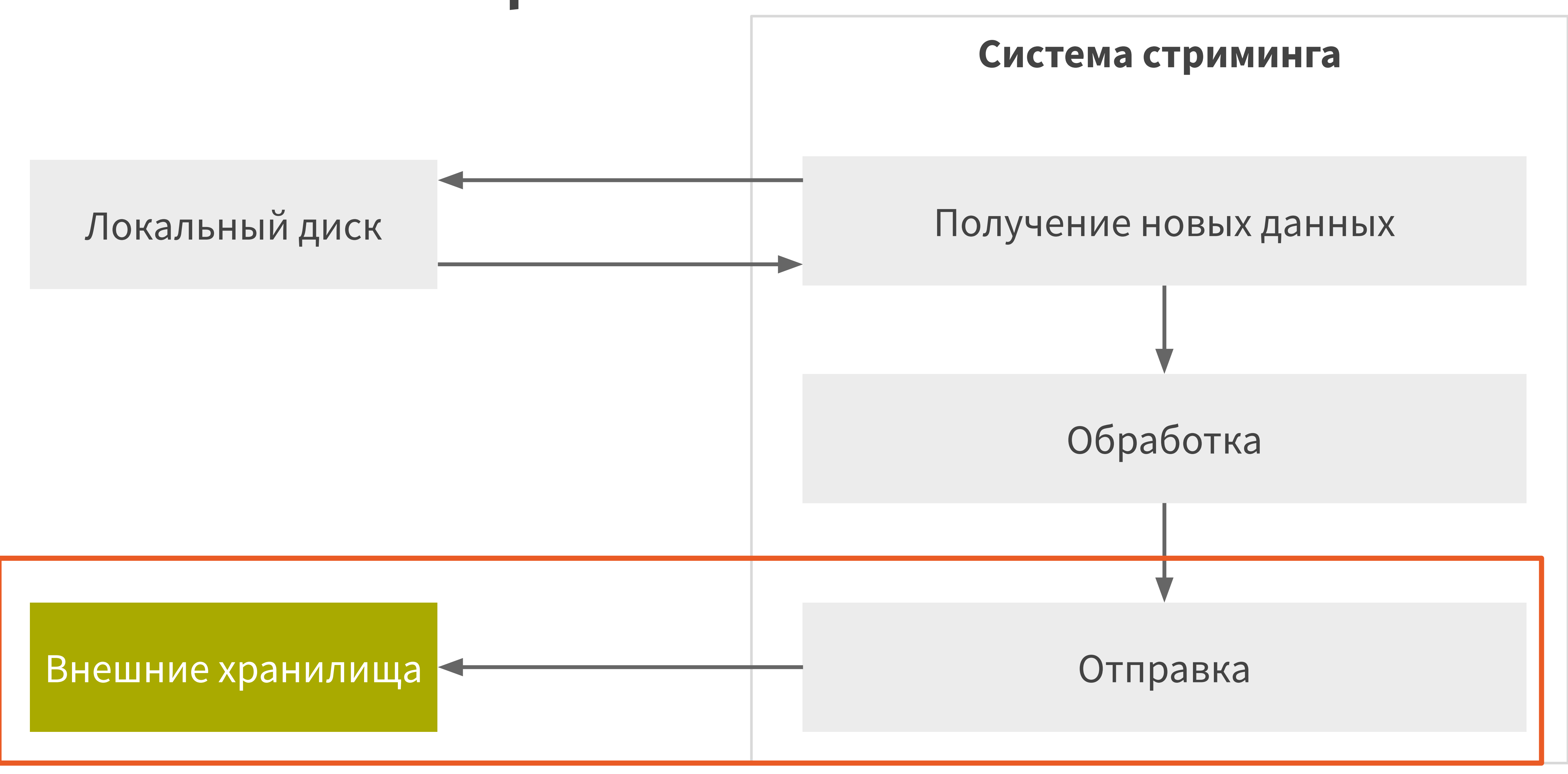


Схема системы стриминга



Схема системы стриминга



Система стриминга: Файловое I/O

Мониторинг

Локальный диск

Получение новых данных

Чтение

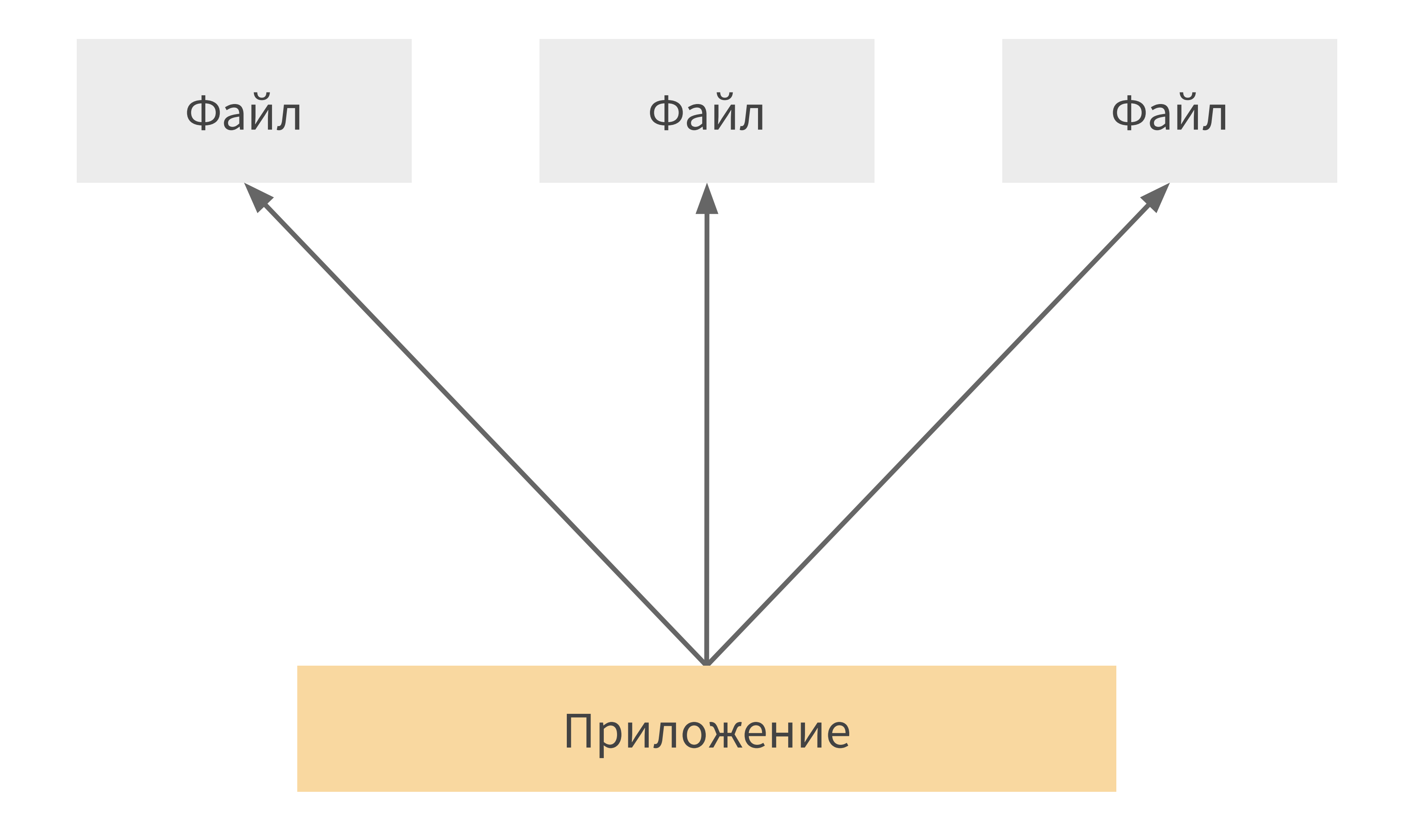
Система стриминга: Файловое I/O

- Определение изменившихся файлов
- Определение предыдущей позиции стриминга
- Чтение новых данных

Система стриминга: Файловое I/O

- **Определение изменившихся файлов**
- Определение предыдущей позиции стриминга
- Чтение новых данных

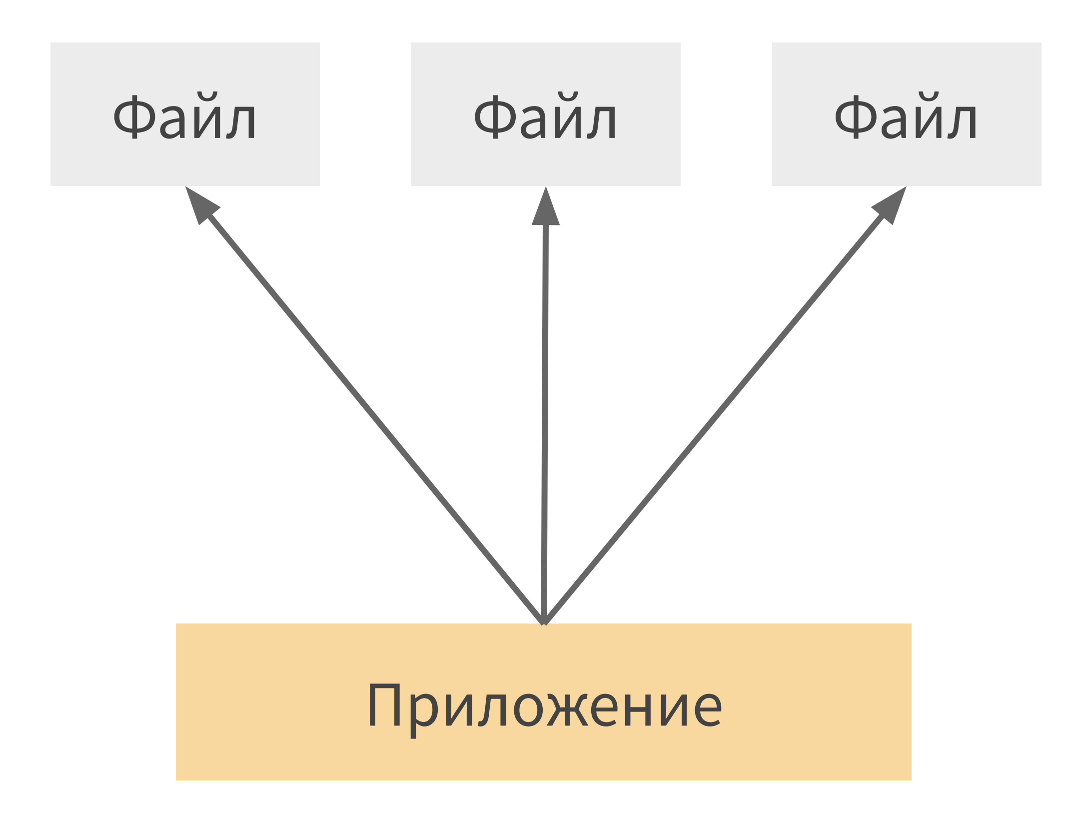
Поллинг



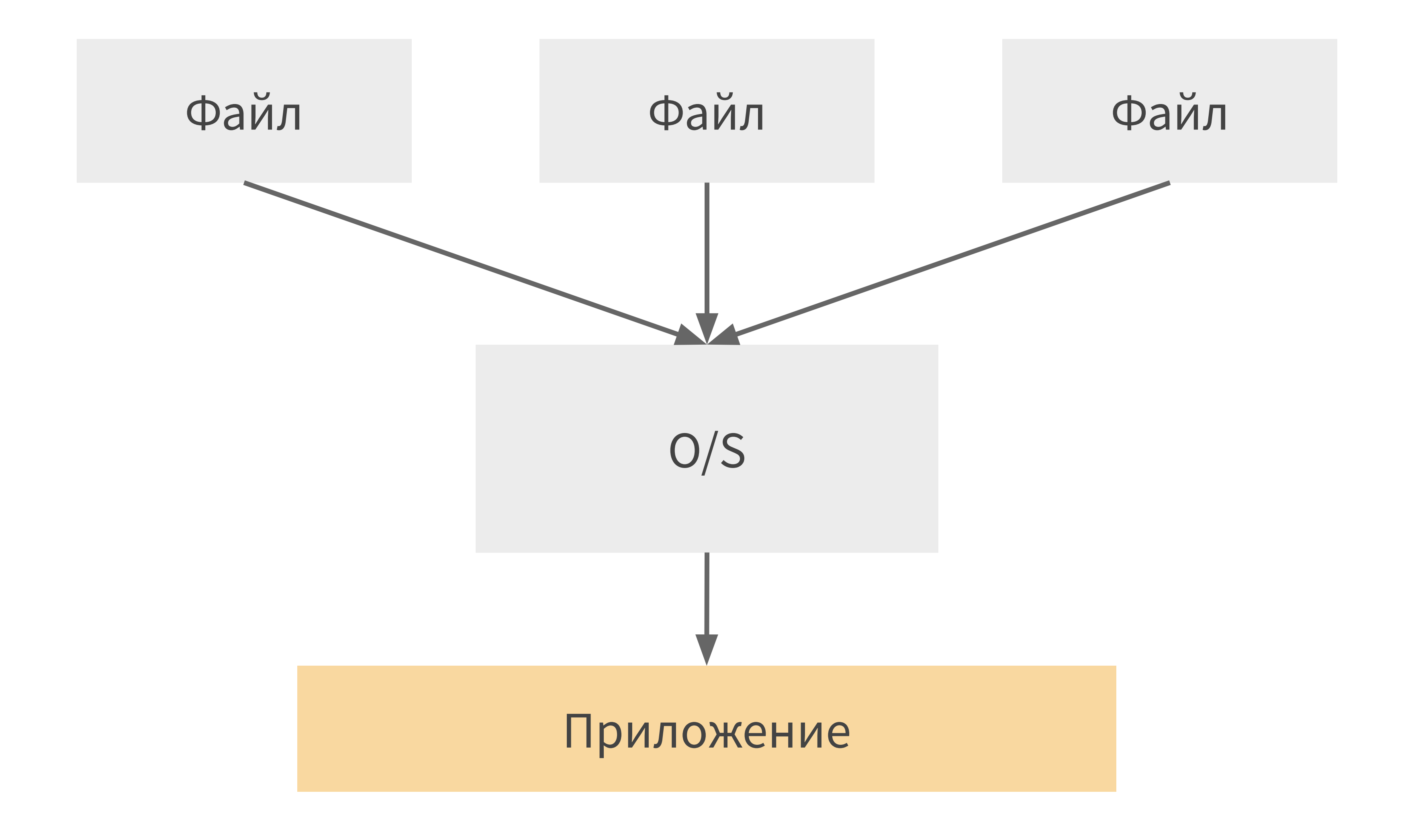
Поллинг

Недостатки:

- Poll с фиксированным интервалом
- Затраты CPU на неизменившиеся файлы



Получение уведомлений O/S

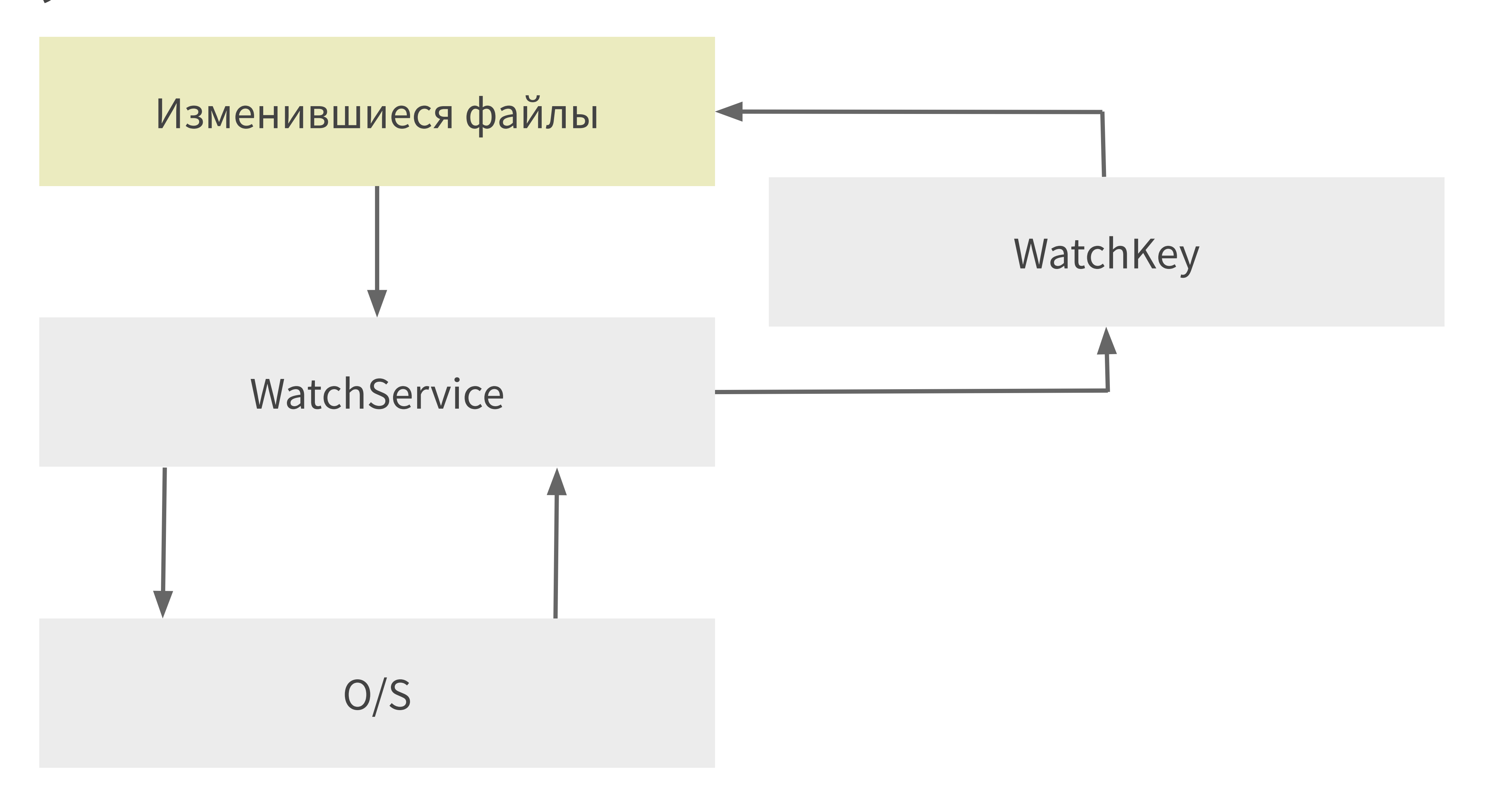


Получение уведомлений O/S: JDK java.nio.file

```
package java.nio.file;

public interface WatchService {
    WatchKey poll(long timeout, TimeUnit unit)
}
...
public interface WatchKey {
    List<WatchEvent<?>> pollEvents();
}
...
public interface WatchEvent<T> {
    T context();
}
```

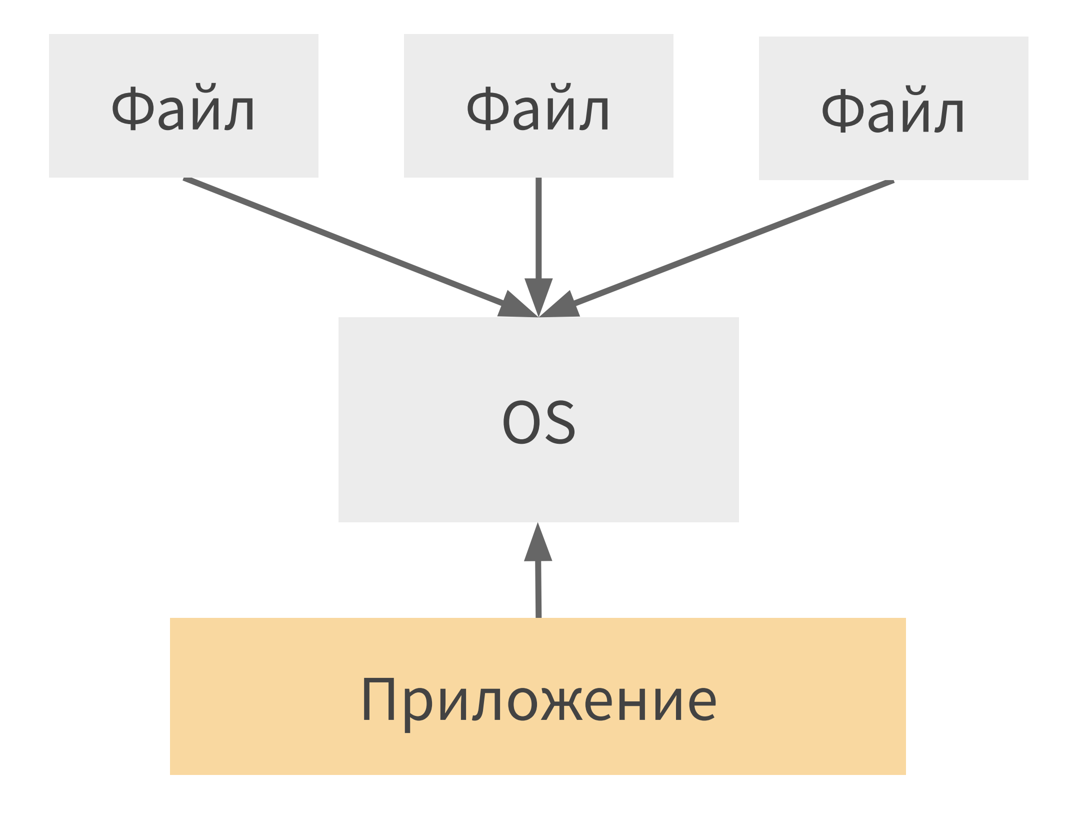
JDK java.nio.file: схема использования



Получение уведомлений O/S: JDK java.nio.file

Недостатки:

- Скудное множество доступных эвентов



JDK java.nio.file: доступные события

```
package java.nio.file;

public final class StandardWatchEventKinds {

    public static final WatchEvent.Kind<Path>   ENTRY_CREATE

    public static final WatchEvent.Kind<Path>   ENTRY_DELETE

    public static final WatchEvent.Kind<Path>   ENTRY_MODIFY

    public static final WatchEvent.Kind<Object> OVERFLOW
}
```

JDK java.nio.file: доступные события

```
package java.nio.file;

public final class StandardWatchEventKinds {

    public static final WatchEvent.Kind<Path>   ENTRY_CREATE

    public static final WatchEvent.Kind<Path>   ENTRY_DELETE

    public static final WatchEvent.Kind<Path>   ENTRY_MODIFY

    public static final WatchEvent.Kind<Object> OVERFLOW
}
```

JDK java.nio.file: доступные события

```java
package java.nio.file;

public final class StandardWatchEventKinds {

    public static final WatchEvent.Kind<Path>   ENTRY_CREATE

    public static final WatchEvent.Kind<Path>   ENTRY_DELETE

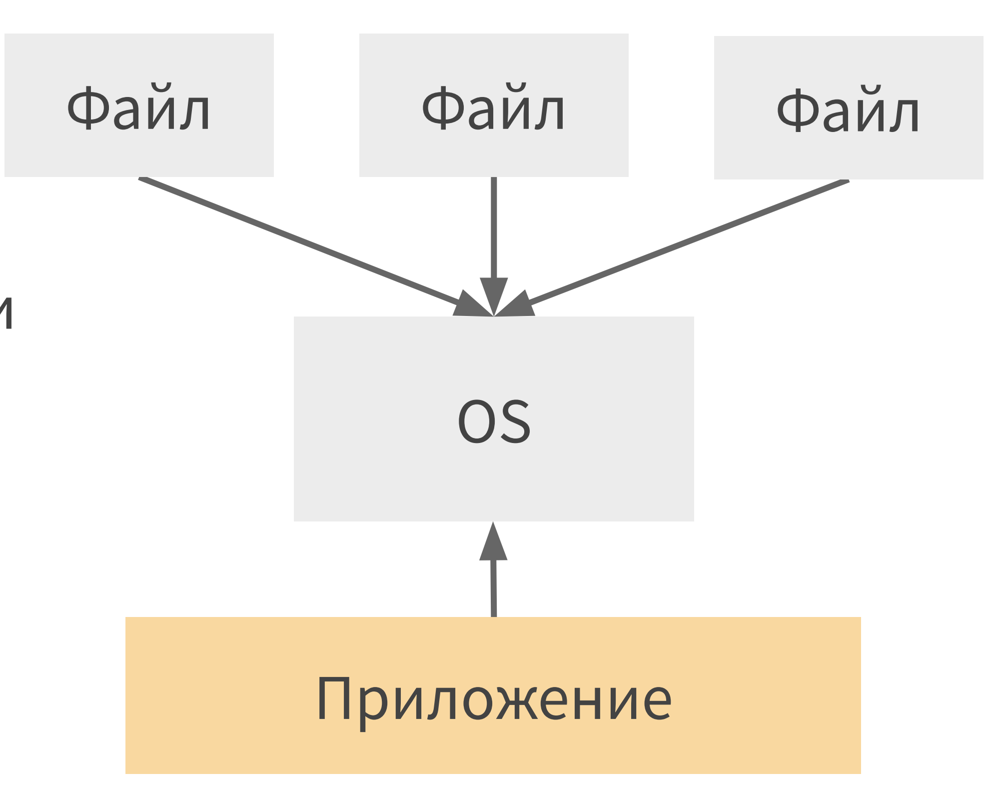
    public static final WatchEvent.Kind<Path>   ENTRY_MODIFY

    public static final WatchEvent.Kind<Object> OVERFLOW
}
```

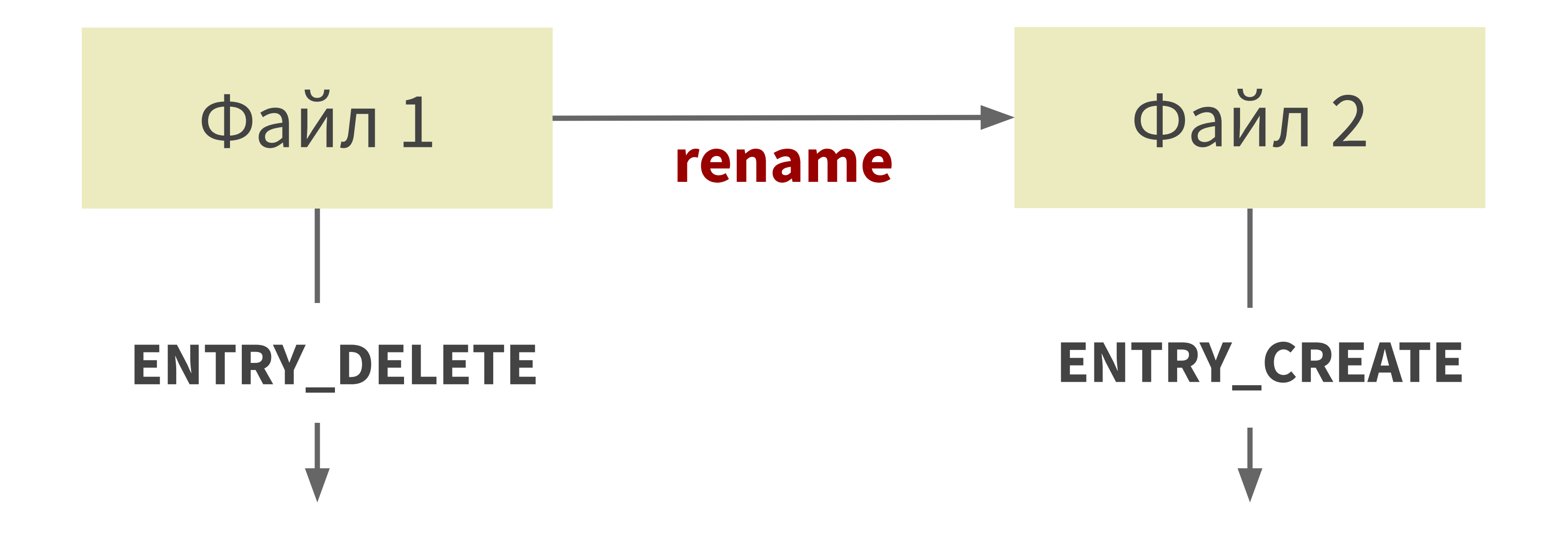
Получение уведомлений O/S: JDK java.nio.file

Недостатки:

- Скудное множество доступных эвентов
- Игнорируется информация о перемещении



JDK java.nio.file: переименование файла



JDK java.nio.file: переименование файла

LinuxWatchService.c *

```
JNIEXPORT jintArray JNICALL
Java_sun_nio_fs_LinuxWatchService_eventOffsets(...)
{
    jint arr[5];
    arr[0] = (jint)offsetof(struct inotify_event, wd);
    arr[1] = (jint)offsetof(struct inotify_event, mask);
    arr[2] = (jint)offsetof(struct inotify_event, cookie);
    arr[3] = (jint)offsetof(struct inotify_event, len);
    arr[4] = (jint)offsetof(struct inotify_event, name);
    ...
}
```

cookie



* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/linux/native/libnio/fs/LinuxWatchService.c#l54

JDK java.nio.file: переименование файла

```
package sun.nio.fs;

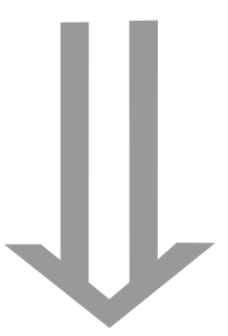
private static class Poller extends AbstractPoller {
    public static final int SIZEOF_INOTIFY_EVENT = eventSize();
    public static final int[] offsets            = eventOffsets();
    public static final int OFFSETOF_WD          = offsets[0];
    public static final int OFFSETOF_MASK        = offsets[1];
    public static final int OFFSETOF_LEN         = offsets[3];
    public static final int OFFSETOF_NAME        = offsets[4];
    ...
}
```

offsets[2]?

Получение уведомлений O/S: JDK java.nio.file

Подмножество событий O/S

Недоступны перемещения



Не подходит при необходимости использования всех эвентов O/S

Получение уведомлений O/S: inotify/Linux

```
#include <sys/inotify.h>

struct inotify_event {
    int      wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char     name[];
}

int inotify_init(void);
int inotify_add_watch(int fd, const char *pathname, uint32_t mask);
int inotify_rm_watch(int fd, int wd);
```



Получение уведомлений O/S: inotify/Linux

```c
#include <sys/inotify.h>

struct inotify_event {
    int      wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char     name[];
}

int inotify_init(void);
int inotify_add_watch(int fd, const char *pathname, uint32_t mask);
int inotify_rm_watch(int fd, int wd);
```



Получение уведомлений O/S: inotify/Linux

```c
#include <sys/inotify.h>

struct inotify_event {
    int      wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char     name[];
}

int inotify_init(void);
int inotify_add_watch(int fd, const char *pathname, uint32_t mask);
int inotify_rm_watch(int fd, int wd);
```



inotify/Linux: доступные события

inotify.h *

```
#define IN_ACCESS        0x00000001
#define IN_MODIFY        0x00000002
#define IN_ATTRIB        0x00000004
#define IN_CLOSE_WRITE   0x00000008
#define IN_CLOSE_NOWRITE 0x00000010
#define IN_OPEN          0x00000020
#define IN_MOVED_FROM    0x00000040
#define IN_MOVED_TO      0x00000080
#define IN_CREATE        0x00000100
#define IN_DELETE        0x00000200
#define IN_DELETE_SELF   0x00000400
#define IN_MOVE_SELF     0x00000800
```



* https://elixir.bootlin.com/linux/v5.3/source/include/uapi/linux/inotify.h#L29

inotify/Linux: доступные события

inotify.h *

```
#define IN_ACCESS        0x00000001
#define IN_MODIFY        0x00000002
#define IN_ATTRIB        0x00000004
#define IN_CLOSE_WRITE   0x00000008
#define IN_CLOSE_NOWRITE 0x00000010
#define IN_OPEN          0x00000020
#define IN_MOVED_FROM    0x00000040
#define IN_MOVED_TO      0x00000080
#define IN_CREATE        0x00000100
#define IN_DELETE        0x00000200
#define IN_DELETE_SELF   0x00000400
#define IN_MOVE_SELF     0x00000800
```



* https://elixir.bootlin.com/linux/v5.3/source/include/uapi/linux/inotify.h#L29

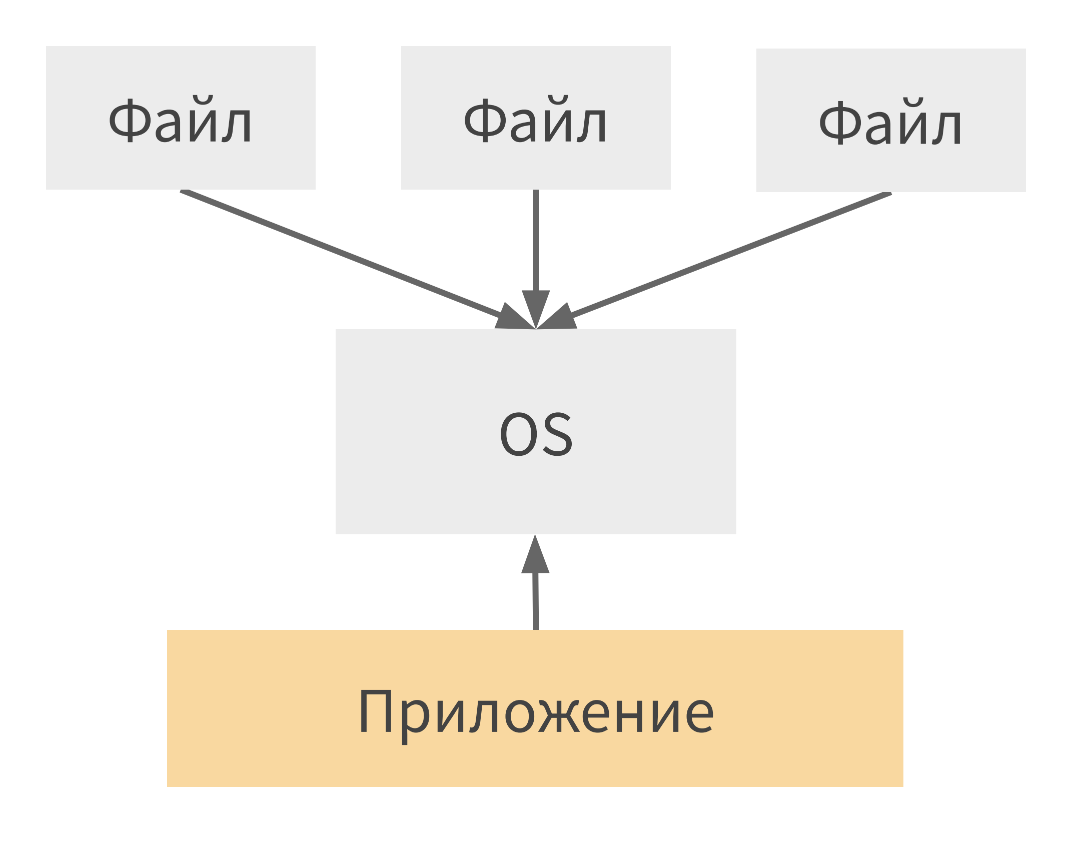
Получение уведомлений O/S: inotify/Linux

Преимущества:

- **Полный набор событий OS Linux**

Недостатки:

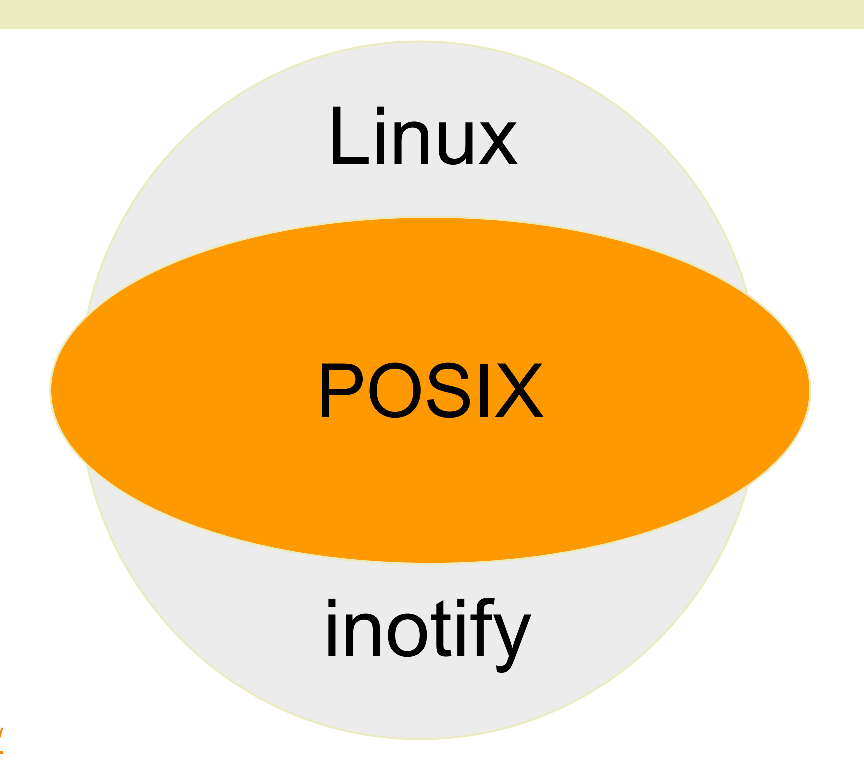
- Не кросс-платформенно / Не POSIX



Стандарт POSIX: описание

Документация POSIX*:

POSIX.1-2017 defines a standard operating system interface and environment, [...] including utility conventions and **C-language header definitions** [...]



* https://pubs.opengroup.org/onlinepubs/9699919799/

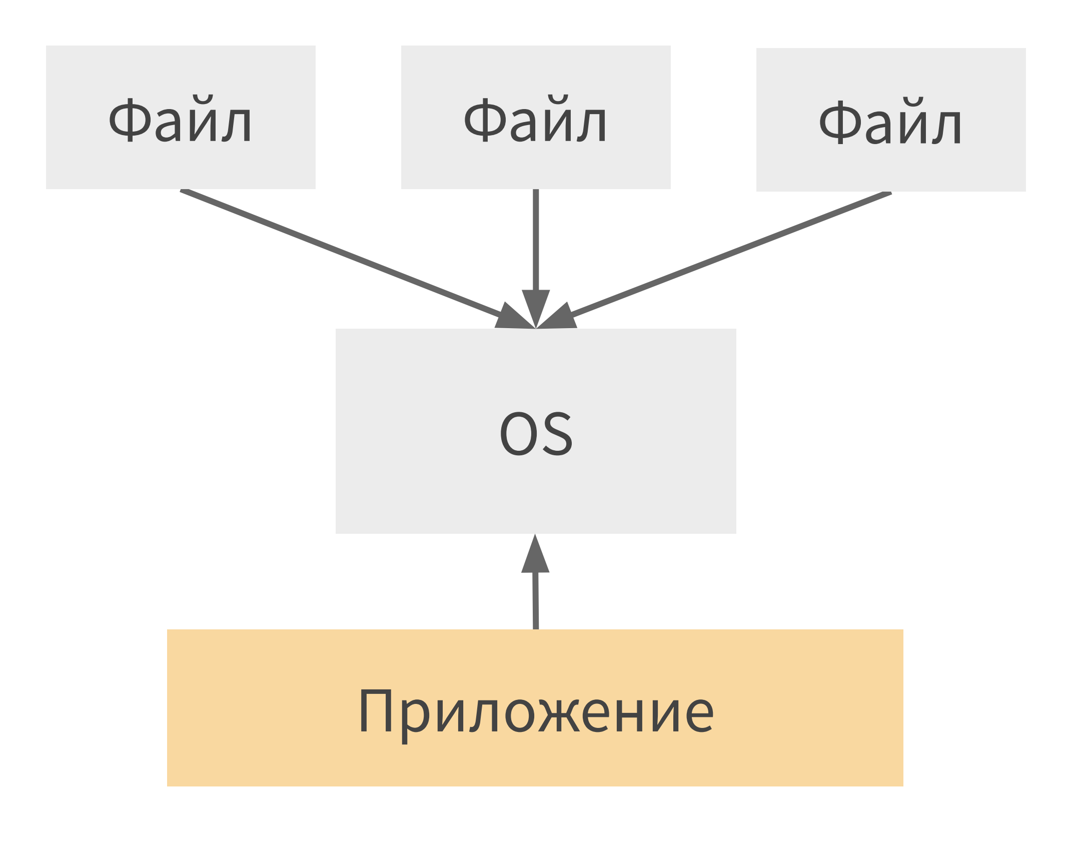
Получение уведомлений O/S: inotify/Linux

Преимущества:

- **Полный набор событий OS Linux**

Недостатки:

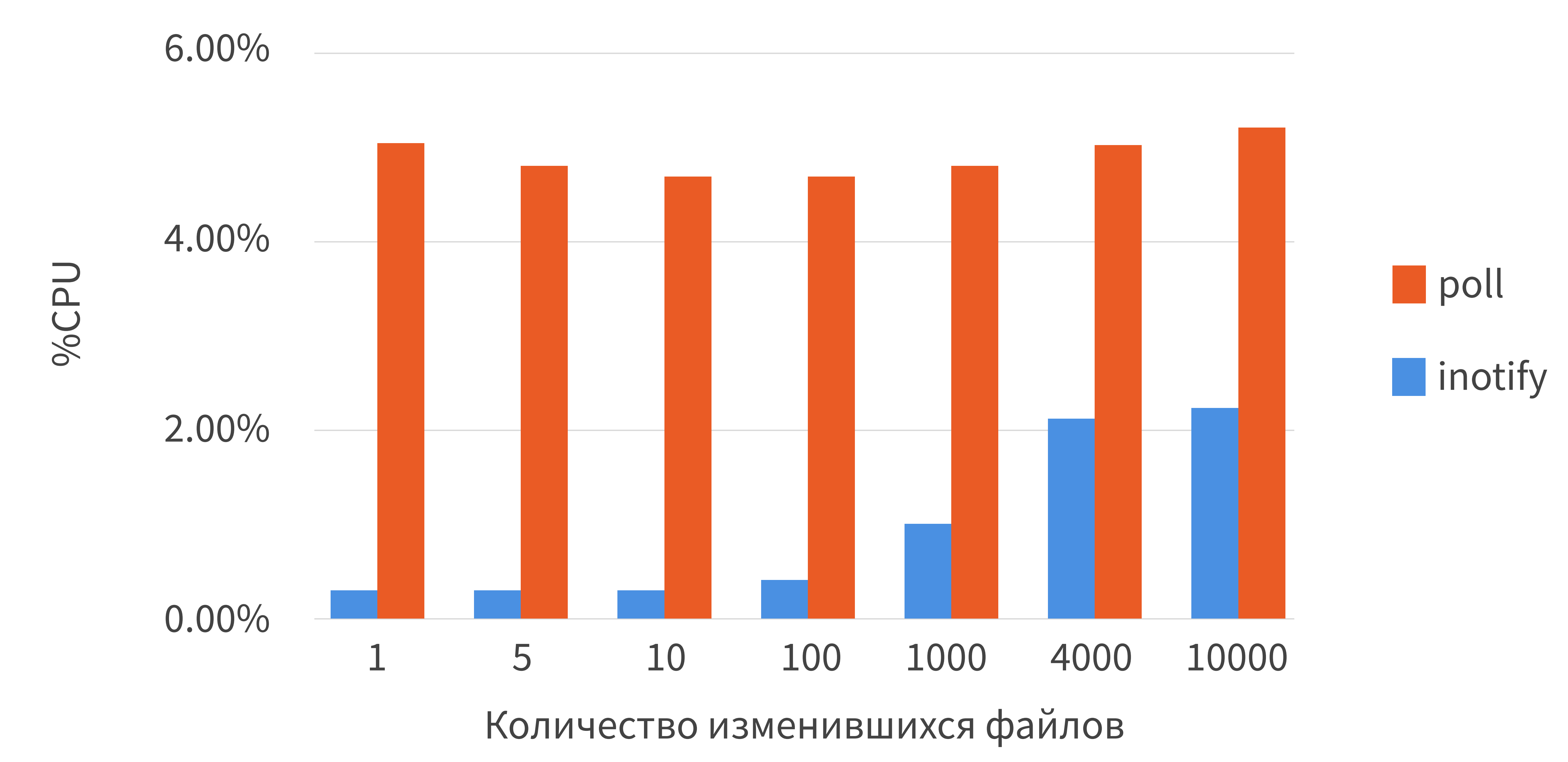
- Не кросс-платформенно / Не POSIX
- Взаимодействие с нативным кодом



Определение изменившихся файлов: перфоманс

- i7-8550U KbL
- Linux Kernel 5.3.0

Потребление CPU: модификации 1 раз в секунду



Потребление CPU: модификации 10 раз в секунду



Определение изменившихся файлов: резюме

- **Поллинг**

  + Простота

  - Негибкость

  - Ресурсоемкость

- **JDK WatchService**

  + Портируемость

  + Только нужные файлы

  - Не все события

- **inotify**

  + Доступны все события

  + Только нужные файлы

  - Не POSIX

  - Нативный код

Система стриминга: файловое I/O

Мониторинг

Локальный диск

inotify

Получение новых данных

Чтение

Система стриминга: Файловое I/O

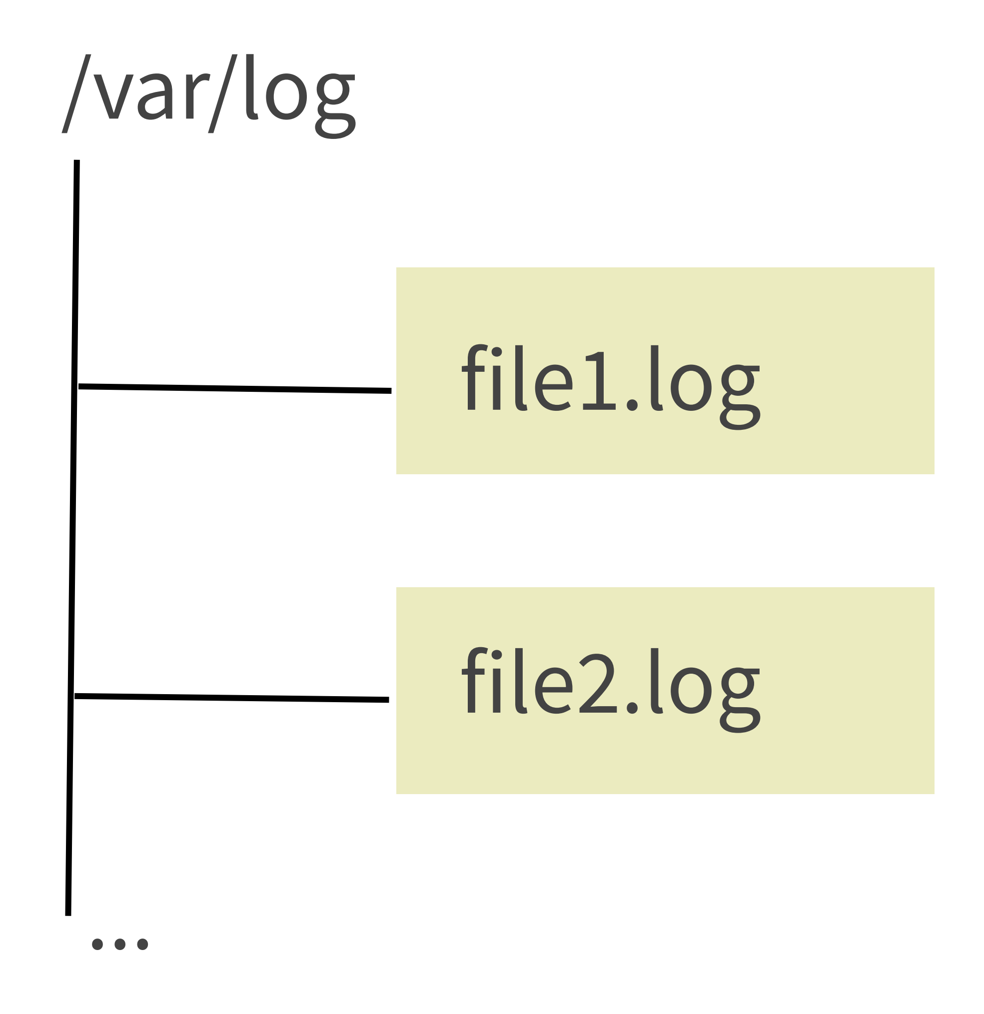
- Определение изменившихся файлов 
- **Определение предыдущей позиции стриминга**
- Чтение новых данных

Определение предыдущей позиции стриминга

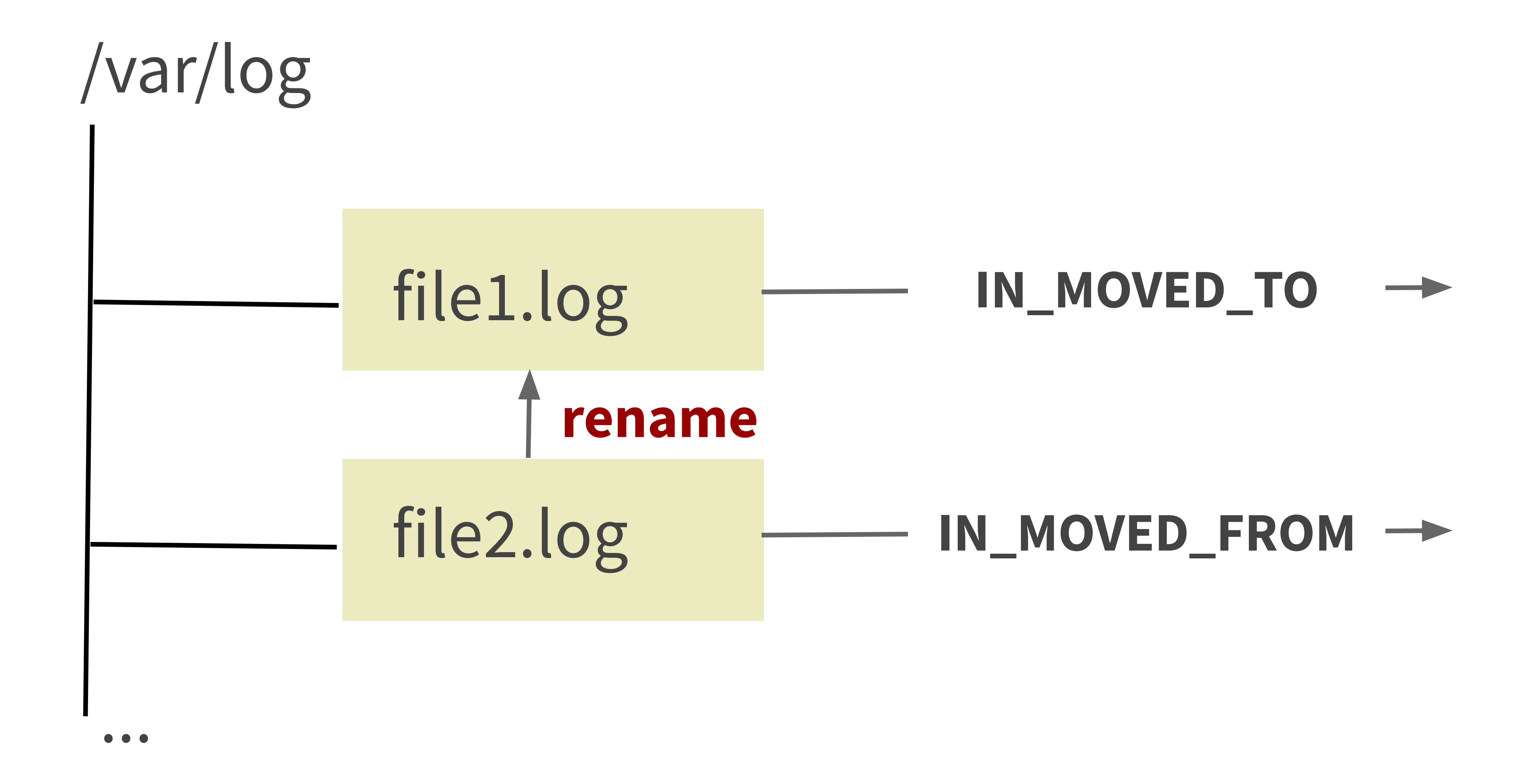
Определение предыдущей позиции

Идентификация файла

Идентификация файла: абсолютный путь



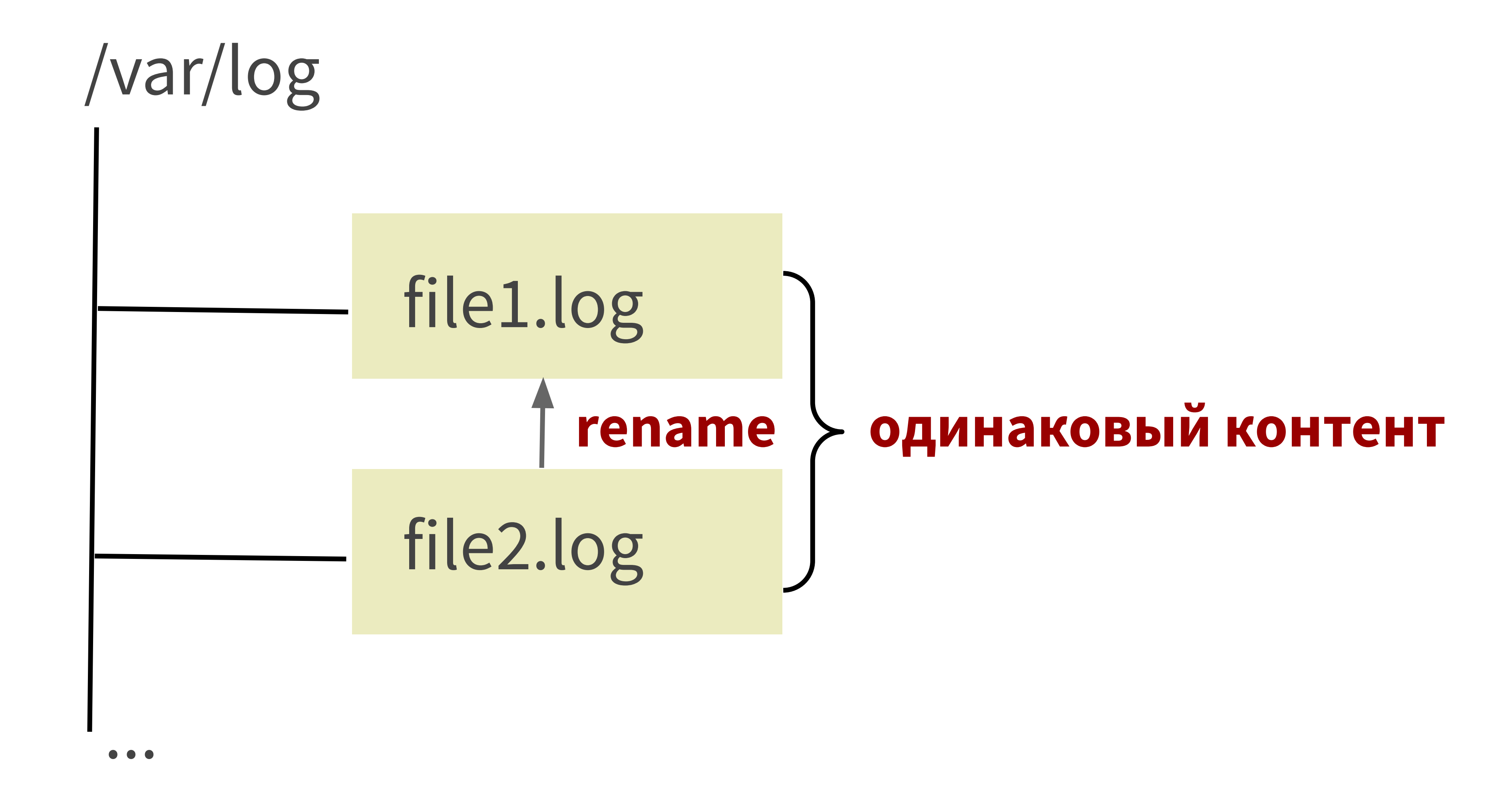
Идентификация файла: абсолютный путь



Идентификация файла: хэш первых n байт



Идентификация файла: хэш первых n байт



Идентификация файла: POSIX file serial number

Документация POSIX*:

3.176 File Serial Number

A per-file system unique identifier for a file.

* https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html

Идентификация файла: POSIX file serial number

Документация POSIX*:

> The <*sys/stat.h*> header shall define the structure of the data returned by the functions *fstat()*, *lstat()*, and *stat()*.

```
struct stat {
    ...
    ino_t st_ino;
    ...
}
```



* 
https://pubs.opengroup.org/onlinepubs/009695399/basedefs/sys/stat.h.html

POSIX file serial number: JDK java.nio.file

```
package java.nio.file;

public final class Files {

    public static Object getAttributes(Path path, String attribute,
                                       LinkOption... options)


}
```

JDK java.nio.file: пример

```
Long inodeNumber = (Long) Files.getAttributes(path, “unix:ino”);
```



POSIX file serial number: JDK java.nio.file

UnixNativeDispatcher.c: *

```
JNIEXPORT void JNICALL
Java_sun_nio_fs_UnixNativeDispatcher_stat0(...){
    ...
    RESTARTABLE(stat64(path, &buf), err);
    ...
}
```

stat64 = stat на x86_64



* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/fs/UnixNativeDispatcher.c#l498

POSIX file serial number: JDK java.nio.file

Недостатки:

- Привязка к POSIX
- **Используется путь к файлу**

JDK java.nio.file: Race Condition

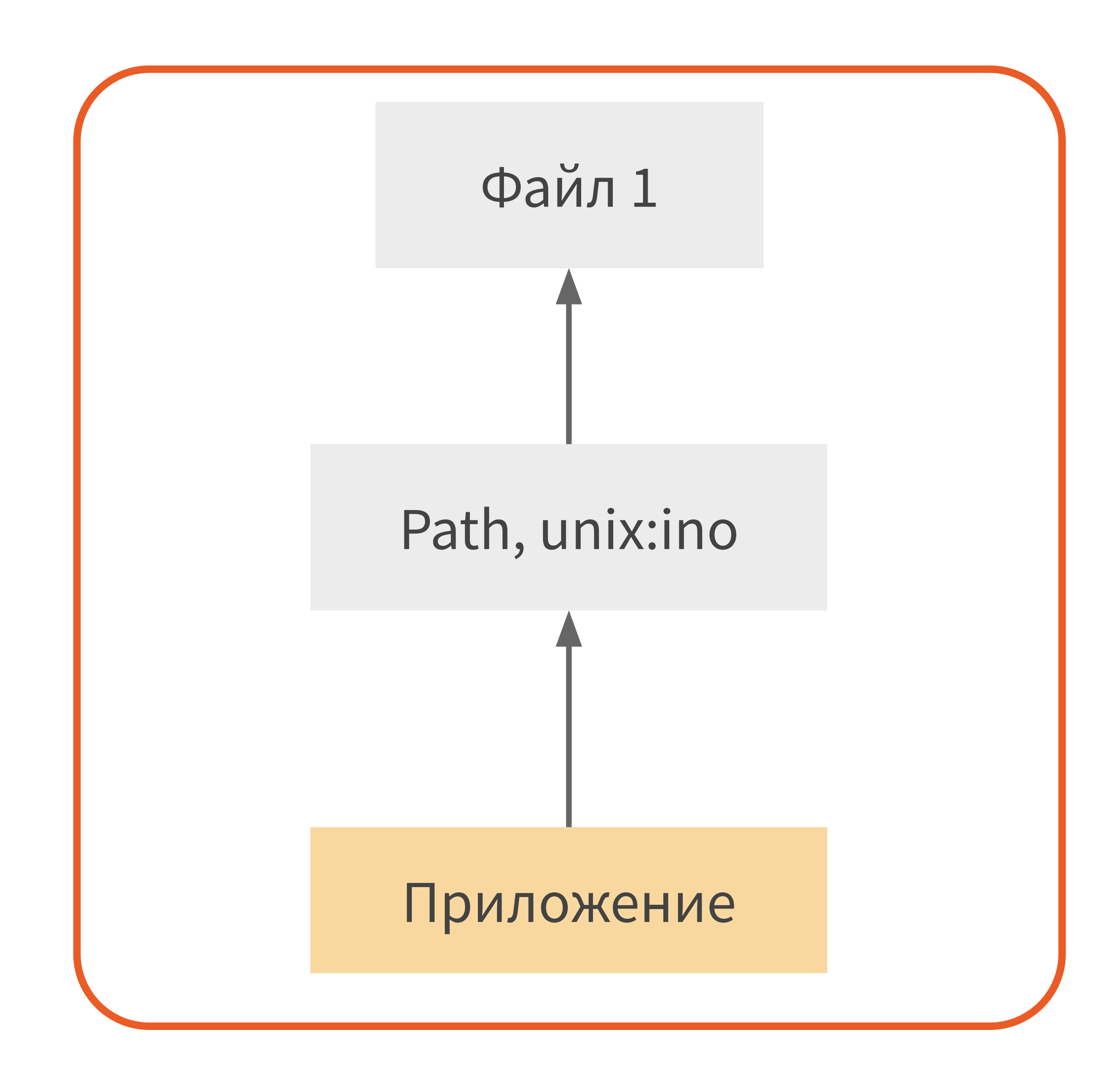
Файл 2

Файл 1

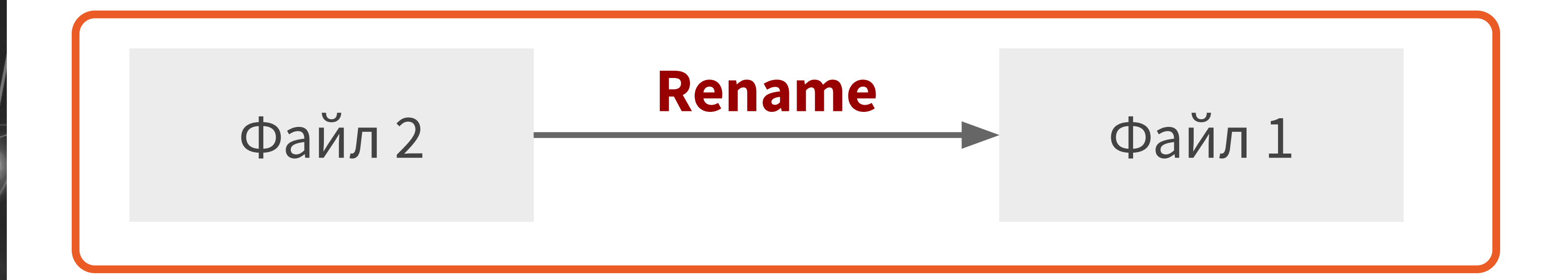
Запись в Файл 1

JDK java.nio.file: Race Condition

Файл 2



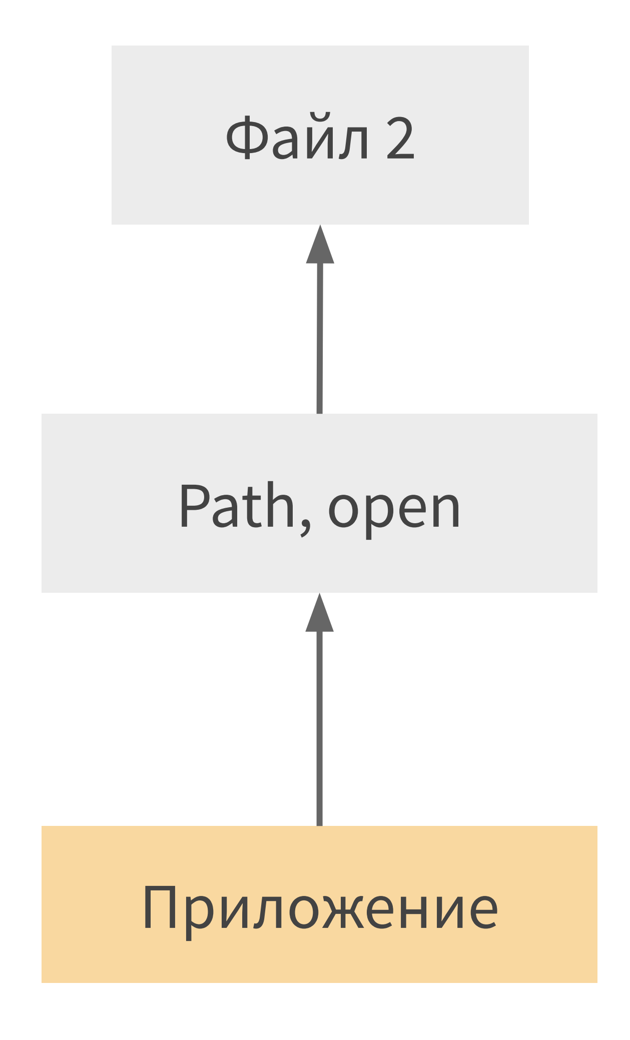
JDK java.nio.file: Race Condition



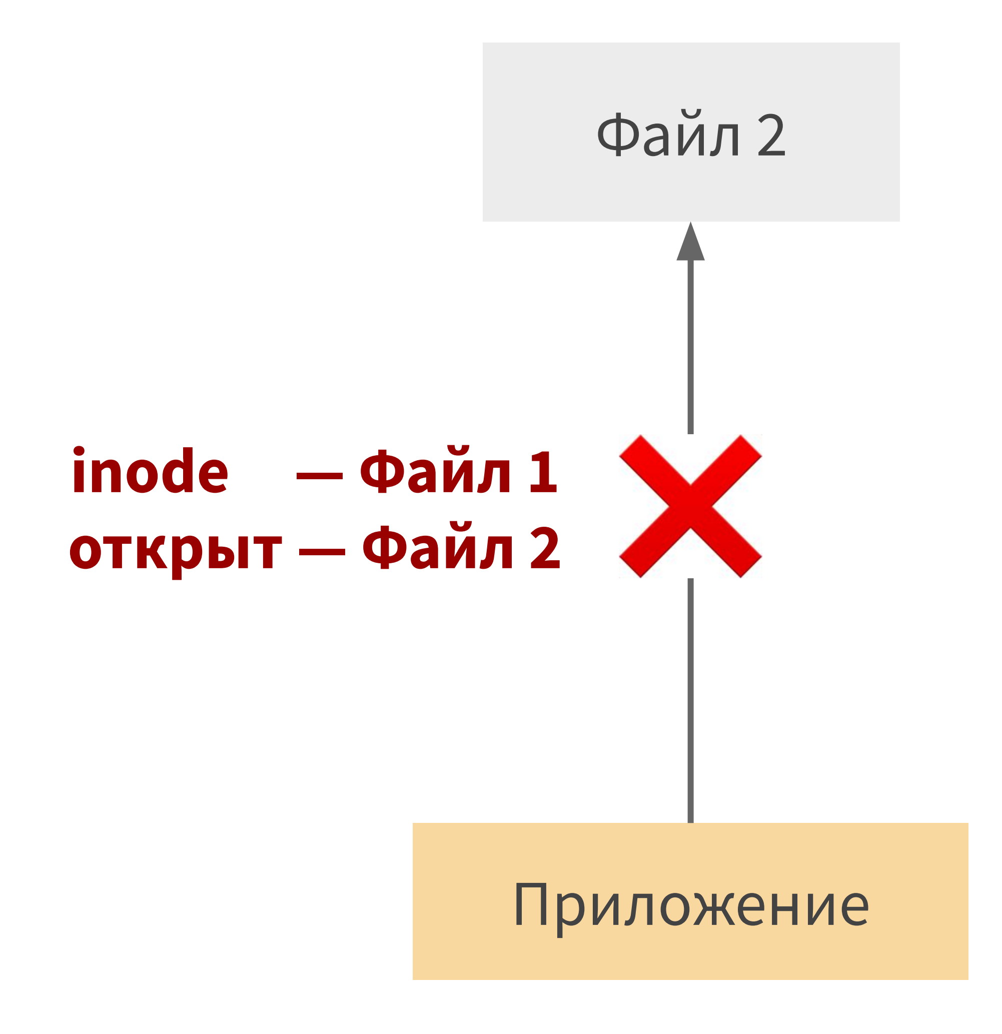
Приложение



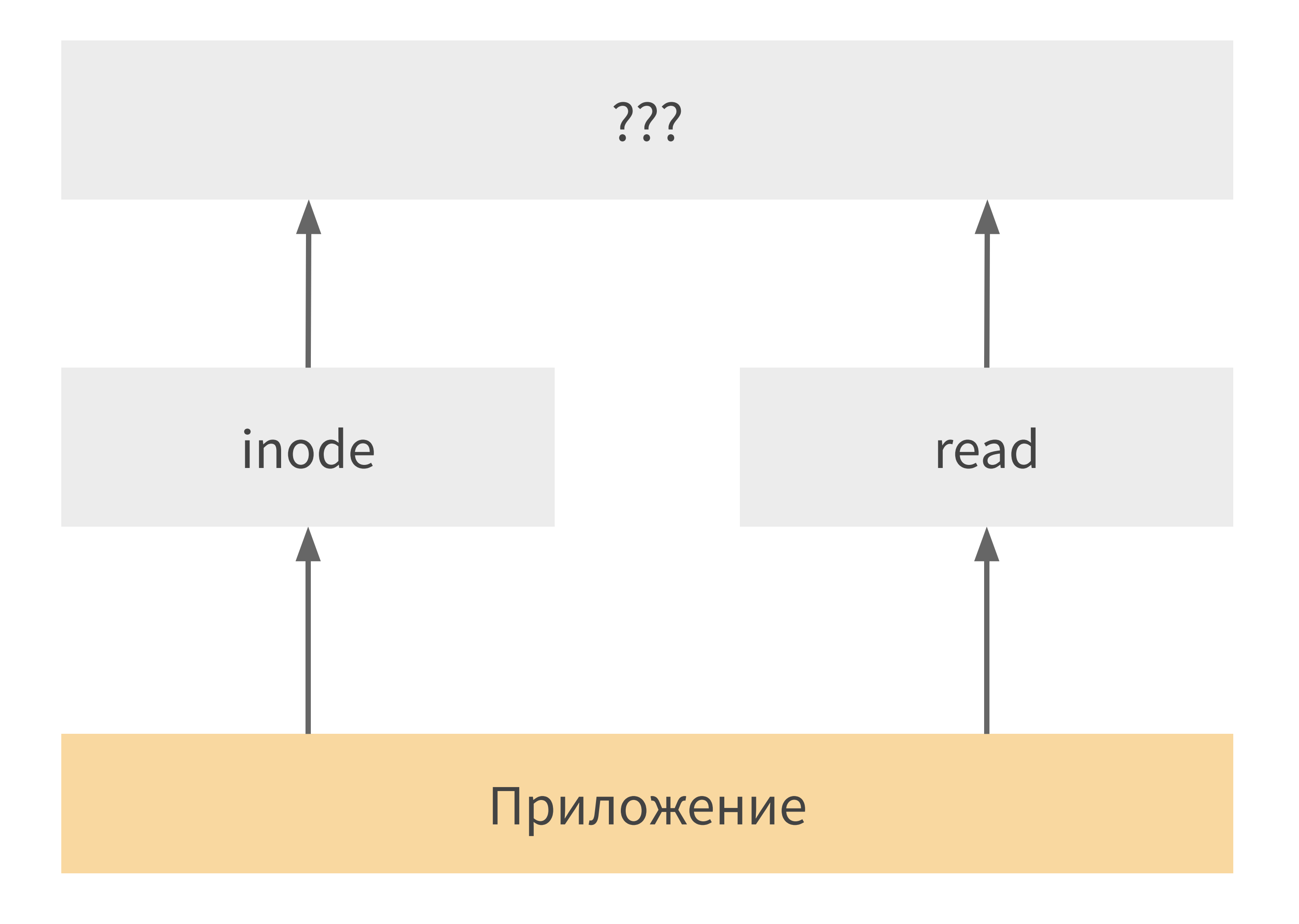
JDK java.nio.file: Race Condition



JDK java.nio.file: Race Condition



Идентификация файла: Race-Free way



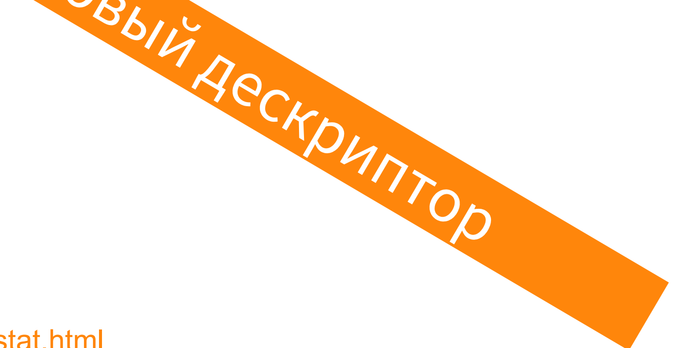
POSIX file serial number: POSIX fstat

Документация POSIX*:

```c
#include <sys/stat.h>

int stat(const char *pathname, ...);

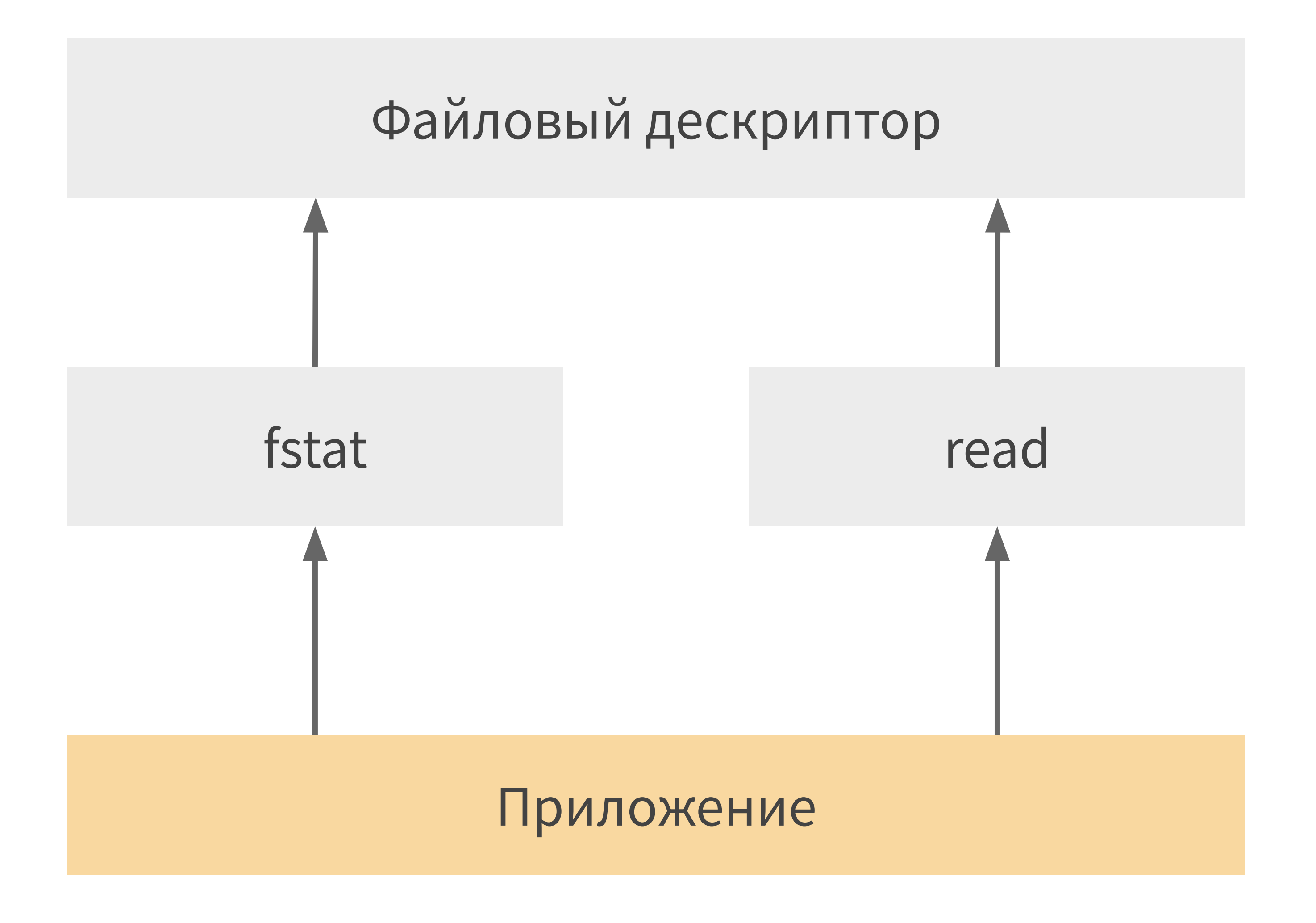
int fstat(int fd, ...);
```



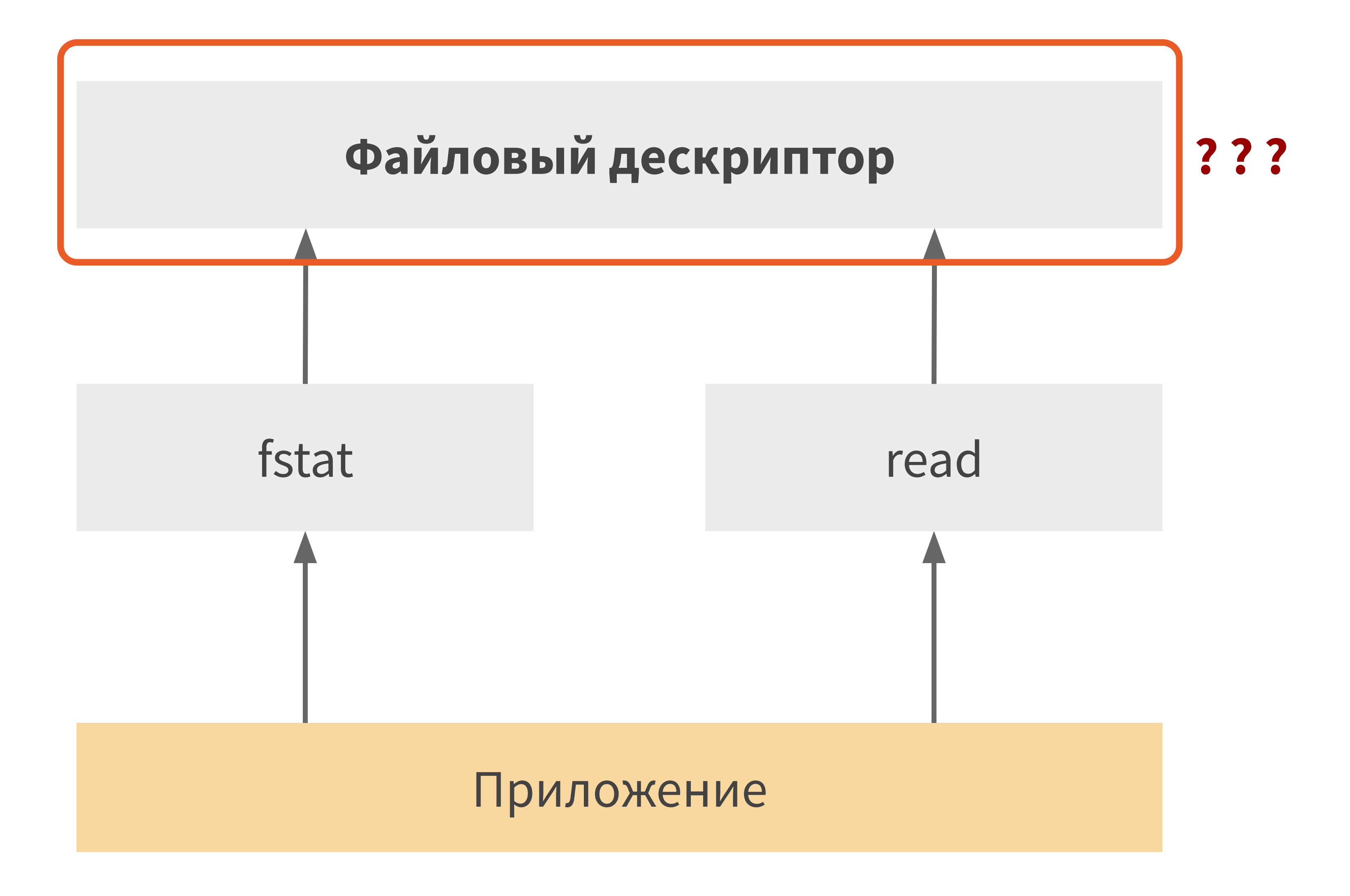
* https://pubs.opengroup.org/onlinepubs/009695399/functions/fstat.html



Идентификация файла: Race-Free way



Идентификация файла: Race-Free way



POSIX fstat: файловый дескриптор

Приватное поле:

```
package java.io;

public final class FileDescriptor {
    private int fd;
}
```

POSIX fstat: файловый дескриптор

Реализация в one-nio:

```
package one.nio.os;

public final class Mem {
    private static final Field fdField = ...

    public static int getFd(FileDescriptor fd){
        try {
            return fdField.getInt(fd);
        } catch (IllegalAccessException e) {
            ...
        }
    }
}
```

POSIX fstat: файловый дескриптор

Нативный метод:

```
package com.company;

public class PosixUtils {
    public static native int openRO(String path);
}
```

POSIX fstat: файловый дескриптор

Реализация:

```
JNIEXPORT void JNICALL
Java_com_company_PosixUtils_openRO(...){
    const char *native_path = (*env)->GetStringUTFChars(...);
    int fd = open(native_path, O_RDONLY);
    (*env)->ReleaseStringUTFChars(...);
    return fd;
}
```



POSIX file serial number: POSIX fstat

Преимущества:

- **Отсутствие Race Condition**

Недостатки:

- Требует файловый дескриптор
- Отсутствует в JDK

Идентификация файла: перфоманс

- горячий dentry cache
- горячий page cache

Идентификация файла: перфоманс

```
#include <x86intrin.h>

unsigned long long __rdtsc(void);

#include <immintrin.h>

void _mm_clflush(void const *A);
```



Идентификация файла: перфоманс

```
#include <x86intrin.h>

unsigned long long __rdtsc(void);

#include <immintrin.h>

void _mm_clflush(void const *A);
```

Чтение Core Time-Stamp Counter



Time-Stamp Counter

Intel System Programming Manual/17.17:

the time-stamp counter increments at a constant rate[...]

The invariant TSC will run at a constant rate in all ACPI P-, C-. and T-states.

Идентификация файла: перфоманс

```
#include <x86intrin.h>

unsigned long long __rdtsc(void);

#include <immintrin.h>

void _mm_clflush(void const *A);
```

Сброс L1/L2/LLC/… по адресу A



Перфоманс: stat vs fstat

	Time-Stamp Counter
stat	2010
fstat	1326

40%

Лучше



Перфоманс: stat vs fstat

```
-    42,76%     0,28%  bin         [kernel.kallsyms]  [k] user_path_at_empty
   - 42,48% user_path_at_empty
      - 30,21% filename_lookup
         - 28,39% path_lookupat
            - 20,55% link_path_walk.part.33
               + 9,64% inode_permission
               + 7,29% walk_component
            + 2,49% complete_walk
            + 2,39% walk_component
            + 1,72% path_init
         + 0,97% putname
      + 11,96% getname_flags
```

Обход каждого из компонентов пути

Перфоманс: stat vs open fstat

Time-Stamp Counter

4000
3000
2000
1000
0

	stat	open fstat
Time-Stamp Counter	2010	4265

110%

Лучше

Перфоманс: pread vs open fstat, чистый кэш

	Time-Stamp Counter
pread 8KB	10282
open fstat	4669

120%

Лучше



Идентификация файла: резюме

● Абсолютный путь	● Хэш первых n байт	● Inode number
+ Доступен в inotify	+ Портируемость	+ Уникален на FS + Перфоманс
- Не идентифицирует	- Не идентифицирует - Перфоманс	- Часть POSIX API

Идентификация файла: резюме

- **JDK Files.getAttribute**

+ Часть JDK

- Race Condition при использовании пути к файлу

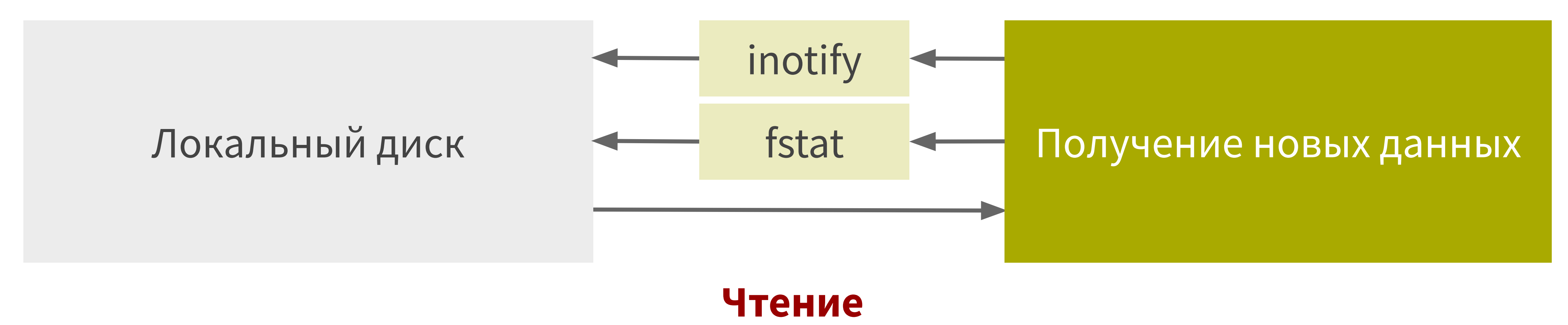
- **POSIX fstat**

+ Не подвержен Race Condition

- Требует файловый дескриптор

- Недоступен в JDK

Система стриминга: Файловое I/O



Система стриминга: Файловое I/O

- Определение изменившихся файлов 
- Определение предыдущей позиции стриминга 
- **Чтение новых данных**

Чтение новых данных

- Пропускная способность
- Использование on-heap byte[]
  - GZIP до JDK11
  - GCP Connector

Чтение новых данных

Zero-copy file

FileChannel.read

java.io.InputStream

Чтение новых данных

Zero-copy file

Чтение новых данных: Zero-copy file

Схема API:

```
package java.nio.channels;

public abstract class FileChannel {

    public abstract long transferTo(long position, long count,
                                    WritableByteChannel target)

}
```

Чтение новых данных: Zero-copy file

FileChannelImpl.c: *

```
JNIEXPORT void JNICALL
Java_sun_nio_ch_FileChannelImpl_transferTo0(...)
{
    ...
    #if defined(__linux__)
        jlong n = sendfile64(dstFD, srcFD, &offset, (size_t)count);
}
```

Zero-copy file



* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/ch/FileChannelImpl.c#l124

Чтение новых данных

FileChannel.read

Чтение новых данных: FileChannel.read

Схема API:

```
package java.nio.channels;

public abstract class FileChannel {

    public abstract int read(ByteBuffer dst, long position)

}
```

Чтение новых данных: FileChannel.read

FileDispatcherImpl.c: *

```c
JNIEXPORT void JNICALL
Java_sun_nio_ch_FileDispatcherImpl_pread0(...)
{
  jint fd = fdval(env, fdo);
  void *buf = (void *)jlong_to_ptr(address);

  return convertReturnVal(env, pread64(fd, buf, len, offset), ...);
}
```

чтение по смещению




* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/unix/native/libnio/ch/FileDispatcherImpl.c#l88

Чтение новых данных: FileChannel.read

IOUtil.java: *

```
package sun.nio.ch;

public class IOUtil {
   static int read(ByteBuffer dst, ...){
       if(dst instanceOf DirectBuffer)
           return readIntoNativeBuffer(...);

      ByteBuffer bb;
      bb = Util.getTemporaryDirectBuffer(...);
      int n = readIntoNativeBuffer(bb, ...);
      ...
   }
}
```

* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/share/classes/sun/nio/ch/IOUtil.java#l226; часть нерелевантного кода опущена

Чтение новых данных: FileChannel.read

IOUtil.java: *

```
package sun.nio.ch;

public class IOUtil {
    static int read(ByteBuffer dst, ...){
         if(dst instanceOf DirectBuffer)
              return readIntoNativeBuffer(...);

         ByteBuffer bb;
         bb = Util.getTemporaryDirectBuffer(...);
         int n = readIntoNativeBuffer(bb, ...);
         ...
    }
}
```

Чтение во временный DirectBuffer

* http://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/share/classes/sun/nio/ch/IOUtil.java#l226

Чтение новых данных

java.io.InputStream

Чтение новых данных: java.io.InputStream

Схема API:

```
package java.io;

public abstract class InputStream implements Closeable {
     public int read(byte[] dst, int off,int len)
}
```

Чтение новых данных: java.io.InputStream

io_util.c: *

```
jint readBytes(jint len, ...)
{
    char stackBuf[BUF_SIZE];
    char *buf = NULL;

    if (len == 0) {
         return 0;
    } else if (len > BUF_SIZE) {
        buf = malloc(len);
    } else {
        buf = stackBuf;
    ...
    IO_read(buf, ...)
```

BUF_SIZE = 8192



* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/java.base/share/native/libjava/io_util.c

Чтение новых данных: резюме

- **Zero-copy file**

+ Нет userspace-копии

- Передача без архивации

- **FileChannel.read**

+ Перфоманс в offheap

- Копирование onheap

- **InputStream**

+ Читает в byte[]

- Копирование onheap

- Аллокация в C-heap

Чтение новых данных: резюме

- ???

+ Читает в byte[]

~~- Копирование onheap~~

~~- Аллокация в C heap~~

Критическая секция JNI

```
void * GetPrimitiveArrayCritical(jarray buf, ...);
```



Критическая секция JNI

```
void * GetPrimitiveArrayCritical(jarray buf, ...);

void ReleasePrimitiveArrayCritical(jarray buf, void * carray, ...);
```





Критическая секция JNI: HotSpot VM

jni.cpp: *

```
jni_GetPrimitiveArrayCritical(...) {
    oop a = lock_gc_or_pin_object(...);
    ...
}
```




* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/share/prims/jni.cpp#l3174

Критическая секция JNI: HotSpot VM

jni.cpp: *

```
static oop lock_gc_or_pin_object(JavaThread* thread, jobject obj){
    if(Universe::heap()->supports_object_pinning()){
        const oop o = JNIHandles::resolve_non_null(obj);
        return Universe::heap()->pin_object(thread, o);
    } else {
        GCLocker::lock_critical(thread);
    ...
}
```



* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/share/prims/jni.cpp#l3155

Критическая секция JNI: HotSpot VM

	supports_object_pinning
Serial	NO
CMS	NO
G1	NO
Parallel Scavenge	NO
Shenandoah	YES
Z	NO

Критическая секция JNI: нативный Java метод

```
package com.company;

public class PosixUtils {
    public static native int read(int fd, byte buf[]);
}
```

Файловый дескриптор

Критическая секция JNI: реализация

```c
JNIEXPORT void JNICALL
Java_com_company_PosixUtils_read(JNIEnv *env, jclass jc,
                                  jint fd, jbyteArray buf){
    size_t sz = (size_t) (*env)->GetArrayLength(env, buf);
    void *buf_ptr = (*env)->GetPrimitiveArrayCritical(env, buf, NULL);
    ssize_t bytesRead = read(fd, buf_ptr, sz);
    (*env)->ReleasePrimitiveArrayCritical(env, buf, buf_ptr, 0);
    return (jint) bytesRead;
}
```



Критическая секция JNI: реализация

```c
JNIEXPORT void JNICALL
Java_com_company_PosixUtils_read(JNIEnv *env, jclass jc,
                                 jint fd, jbyteArray buf){
   size_t sz = (size_t) (*env)->GetArrayLength(env, buf);
   void *buf_ptr = (*env)->GetPrimitiveArrayCritical(env, buf, NULL);
   ssize_t bytesRead = read(fd, buf_ptr, sz);
   (*env)->ReleasePrimitiveArrayCritical(env, buf, buf_ptr, 0);
   return (jint) bytesRead;
}
```



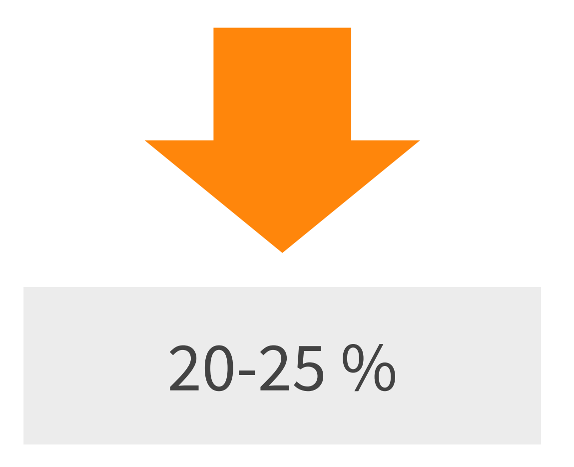
Критическая секция JNI: перфоманс

- горячий page cache

Критическая секция JNI: перфоманс

- горячий page cache

	InputStream	criticalRead
2 GB	0.36 sec	0.27 sec
4 GB	0.59 sec	0.47 sec
6 GB	0.95 sec	0.72 sec



Система стриминга: Файловое I/O



Система стриминга: файловое I/O

- Определение изменившихся файлов 
- Определение предыдущей позиции стриминга 
- Чтение новых данных 

Критическая секция JNI: анализ

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

Критическая секция JNI: анализ

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

```
-    68,44%    68,36%  java              [kernel.kallsyms]   [k] copy_user_enhanced_fast_string
     68,36% 0x7f50ec3e4519
        Java_com_test_PosixUtils_read
        read (inlined)
        __GI___libc_read (inlined)
        entry_SYSCALL_64_after_hwframe
        do_syscall_64
        __x64_sys_read
        ksys_read
        vfs_read
        __vfs_read
        new_sync_read
        ext4_file_read_iter
        generic_file_read_iter
      - copy_page_to_iter
           68,02% copy_user_enhanced_fast_string
```

Критическая секция JNI: анализ

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

```
-   68,44%    68,36%  java            [kernel.kallsyms]   [k] copy_user_enhanced_fast_string
     68,36% 0x7f50ec3e4519
        Java_com_test_PosixUtils_read
        read (inlined)
        __GI___libc_read (inlined)
        entry_SYSCALL_64_after_hwframe
        do_syscall_64
        __x64_sys_read
        ksys_read
        vfs_read
        __vfs_read
        new_sync_read
        ext4_file_read_iter
        generic_file_read_iter
      - copy_page_to_iter
           68,02% copy_user_enhanced_fast_string
```

Копирование из ядра пользователю

Копирование памяти в ядре

copy_user_64.S: *

```
ENTRY(copy_user_enhanced_fast_string)
        ...
        movl %edx,%ecx
1:      rep
        movsb
        xorl %eax,%eax
        ASM_CLAC
        ret
        ...
```

rep movsb

* https://elixir.bootlin.com/linux/v5.3/source/arch/x86/lib/copy_user_64.S#L200

Копирование памяти в ядре

- Почему не используется AVX?
- Будет ли AVX эффективнее `rep movsb`?
- Если AVX быстрее, то можно ли его использовать?

Копирование памяти в ядре

- **Почему не используется AVX?**
- Будет ли AVX эффективнее `rep movsb`?
- Если AVX быстрее, то можно ли его использовать?

Копирование памяти в ядре: SysV ABI

Спецификация AMD64 SysV ABI: *

A system-call is done via the syscall instruction. The kernel destroys registers **%rcx** and **%r11**.

* https://github.com/hjl-tools/x86-psABI/wiki/X86-psABI

System Call: сохранение и восстановление состояния

calling.h: *

```
.macro PUSH_AND_CLEAR_REGS ...
    pushq %rsi
```

```
.macro POP_REGS ...
    popq %rsi
```

* https://elixir.bootlin.com/linux/latest/source/arch/x86/entry/calling.h

System Call: сохранение и восстановление состояния

calling.h: *

```
.macro PUSH_AND_CLEAR_REGS ...
    pushq %rsi
    pushq %rdi
```

```
.macro POP_REGS ...
    popq %rdi
    popq %rsi
```

* https://elixir.bootlin.com/linux/latest/source/arch/x86/entry/calling.h



System Call: сохранение и восстановление состояния

calling.h: *

```
.macro PUSH_AND_CLEAR_REGS ...
    pushq %rsi
    pushq %rdi
    pushq %rdx
```

```
.macro POP_REGS ...
    popq %rdx
    popq %rdi
    popq %rsi
```

* https://elixir.bootlin.com/linux/latest/source/arch/x86/entry/calling.h ;аргумент \rdx развернут в %rdx для удобства чтения

System Call: сохранение и восстановление состояния

calling.h: *

```
.macro PUSH_AND_CLEAR_REGS ...
    pushq %rsi
    pushq %rdi
    pushq %rdx
    ...
    pushq %r13
    pushq %r14
    pushq %r15
```

```
.macro POP_REGS ...
    popq %r15
    popq %r14
    popq %r13
    ...
    popq %rdx
    popq %rdi
    popq %rsi
```

* https://elixir.bootlin.com/linux/latest/source/arch/x86/entry/calling.h ;аргумент \rdx развернут в %rdx для удобства чтения

Копирование памяти в ядре

- Почему не используется AVX?
- **Будет ли AVX эффективнее `rep movsb`?**
- Если AVX быстрее, то можно ли его использовать?

Сравнение AVX2 и rep movsb

System.arraycopy *:

```
else if (UseAVX == 2) {
    __ vmovdqu(xmm0, Address(..., -56));
    __ vmovdqu(Address(..., -56), xmm0);
    __ vmovdqu(xmm1, Address(..., -24));
    __ vmovdqu(Address(..., -24), xmm1);
}
```

* https://hg.openjdk.java.net/jdk/jdk12/file/06222165c35f/src/hotspot/cpu/x86/stubGenerator_x86_64.cpp#l1263

Сравнение AVX2 и rep movsb

libc-2.27/memcpy *:

```
(gdb) disas __memmove_avx_unaligned_erms
...
<+76>:  mov        %rdx,%rcx
<+79>:  rep movsb %ds:(%rsi),%es:(%rdi)
<+81>:  retq
...
<+178>: cmp        $0x1000,%rdx
<+185>: ja         0x18eafb
```

rep movsb

4КБ и более

Сравнение memcpy и System.arraycopy: Java memcpy

```
public class Memcpy {
    public static native void arrayMemcpy(
                  byte src[], int srcOff,
                  byte dst[], int dstOff,
                  int len
    );
}
```

Сравнение memcpy и System.arraycopy: Java memcpy

```
JNIEXPORT void JNICALL Java_com_company_Memcpy_arrayMemcpy(...){
    ...
}

JNIEXPORT void JNICALL JavaCritical_com_company_Memcpy_arrayMemcpy(...){
    memcpy((char *) dst + dst_off,
           (char *) src + src_off,
           (size_t) len);
}
```

Сравнение memcpy и System.arraycopy

```
@Param({“8192”, “131072”, “8388608”, “16777216”, “33554432”})
public int size;
public byte src[], dst[];

@Benchmark
public void arraycopy(){
    System.arraycopy(src, 0, dst, 0, size);
}

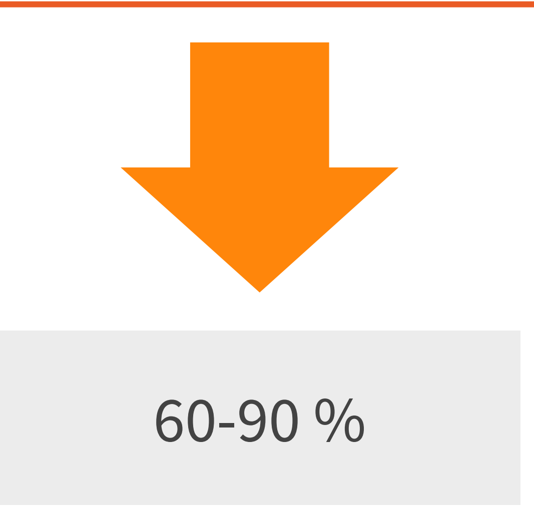
@Benchmark
public void memcpy(){
    Memcpy.arrayMemcpy(src, 0, dst, 0, size);
}
```

Сравнение memcpy и System.arraycopy

	memcpy, μs	System.arraycopy, μs
8 KB	0.172	0.132
128 KB	3.439	4.171
8 MB	542.921	959.101
16 MB	1393.213	2209.657
32 MB	3128.653	4702.299

Сравнение memcpy и System.arraycopy

	memcpy, μs	System.arraycopy, μs
8 KB	0.172	0.132
128 KB	3.439	4.171
8 MB	542.921	959.101
16 MB	1393.213	2209.657
32 MB	3128.653	4702.299



Non-Temporal Stores

```
(gdb) disas __memmove_avx_unaligned_erms
<+43>:    cmp      0x261b76(%rip),%rdx
<+50>:    jae      0x18ec2d
...
<+770>:   vmovntdq %ymm0,(%rdi)
<+774>:   vmovntdq %ymm1,0x20(%rdi)
<+779>:   vmovntdq %ymm2,0x40(%rdi)
<+784>:   vmovntdq %ymm3,0x60(%rdi)
...
<+805>:   sfence
```

x86 shared non-temporal threshold

Non-Temporal Stores: x86 non-temporal threshold

```
(gdb) p __x86_shared_non_temporal_threshold
$1 = 6291356 #6MB
```

Non-Temporal Stores

```
(gdb) disas __memmove_avx_unaligned_erms
<+43>:    cmp    0x261b76(%rip),%rdx
<+50>:    jae    0x18ec2d
...
<+770>:   vmovntdq %ymm0,(%rdi)
<+774>:   vmovntdq %ymm1,0x20(%rdi)
<+779>:   vmovntdq %ymm2,0x40(%rdi)
<+784>:   vmovntdq %ymm3,0x60(%rdi)
...
<+805>:   sfence
```

vmovntdq - Non-Temporal store

Non-Temporal Stores: свойства

Intel Software Optimization Manual/8.4.1:

- Write combining — Successive writes to the same cache line are combined
- Write collapsing — Successive writes to the same bytes(s) result in only the last write being visible
- Weakly ordered — No ordering is preserved between WC stores or between WC stores and other loads or stores
- **Uncacheable and not write-allocating** — Stored data is written around the cache and will not generate a read-for-ownership bus request for the corresponding cache line

Write Back память: regular stores

Regular-запись → L1 Data Cache

Write Back память: regular stores

Regular-запись → L1 Data Cache
MESI: Exclusive/Modified

Write Back память: regular stores

Regular-запись

L1 Data Cache

Read For Ownership

L2/LLC

Memory

Write Back память: non-temporal stores

Non-Temporal-запись

L1 Data Cache

Read For Ownership

L2/LLC

Write-Combining Buffer

Memory

Расположен в CPU-ядре

Libc memcpy: техника копирования

- **4KB до 6 MB**

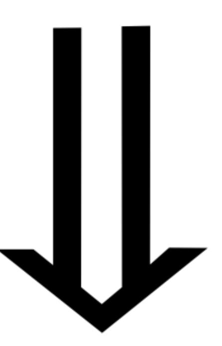
`rep movsb`

- **Более 6MB**

`Non-Temporal stores`

Non-temporal stores: перфоманс копирования

Нет RFO-трафика

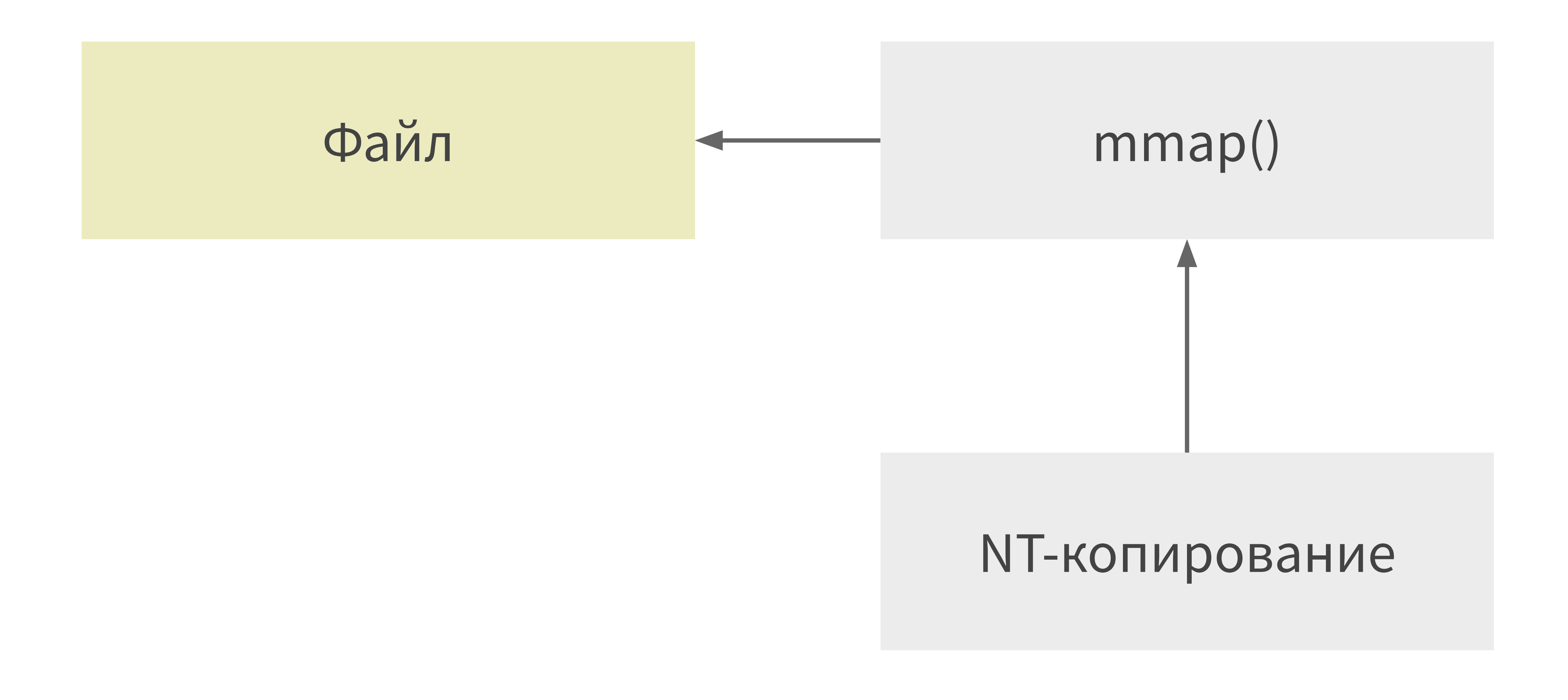


Перфоманс копирования больших данных

Копирование памяти в ядре

- Почему не используется AVX?
- Будет ли AVX эффективнее `rep movsb`?
- **Если AVX быстрее, то можно ли его использовать?**

Чтение из файла: Non-Temporal stores + mmap



mmap: библиотека one-nio

```
String FILE_NAME;
long mappedRegionSize;

one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
        FILE_NAME,
        mappedRegionSize,
        one.nio.mem.MappedFile.MAP_RO
);

mf.close();
```

mmap: библиотека one-nio

```
String FILE_NAME;
long mappedRegionSize;

one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
      FILE_NAME,
      mappedRegionSize,
      one.nio.mem.MappedFile.MAP_RO
);

mf.close();
```

mmap()

mmap: библиотека one-nio

```
String FILE_NAME;
long mappedRegionSize;

one.nio.mem.MappedFile mf = new one.nio.mem.MappedFile(
        FILE_NAME,
        mappedRegionSize,
        one.nio.mem.MappedFile.MAP_RO
);

mf.close();
```

munmap()

Non-Temporal Stores + mmap: перфоманс

- горячий page cache

Non-Temporal Stores + mmap: перфоманс

- горячий page cache

Без учета mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

Non-Temporal Stores + mmap: перфоманс

- горячий page cache

Без учета mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

С учетом mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.82	0.72

Non-Temporal Stores + mmap: анализ

```
$ sudo perf record --call-graph dwarf -F 9123 -p <app_pid>
```

```
- 29,50% __memmove_avx_unaligned_erms
   - 22,56% page_fault
      - 22,51% do_page_fault
         + 22,46% __do_page_fault
   + 3,84% swapgs_restore_regs_and_return_to_usermode
   + 1,79% error_entry
  2,64% page_fault
```

Page Fault при первом доступе

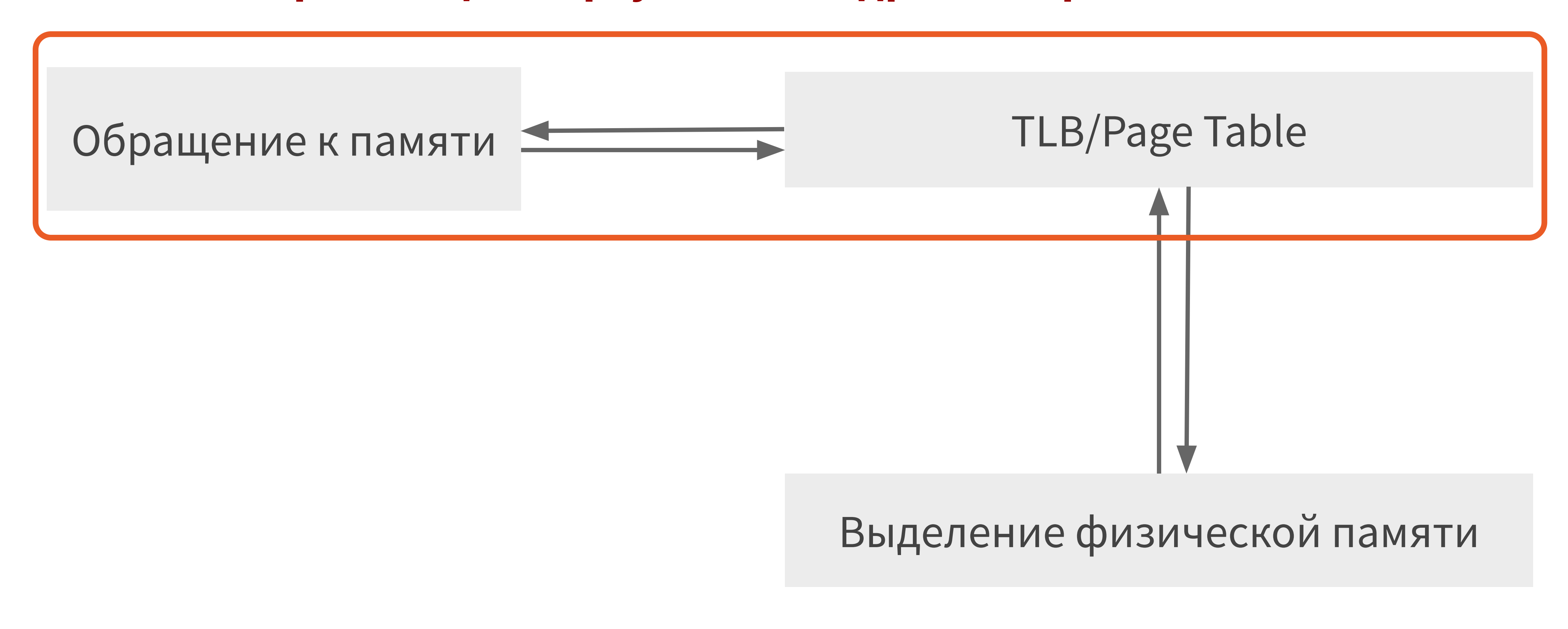
Page Fault

Обращение к памяти

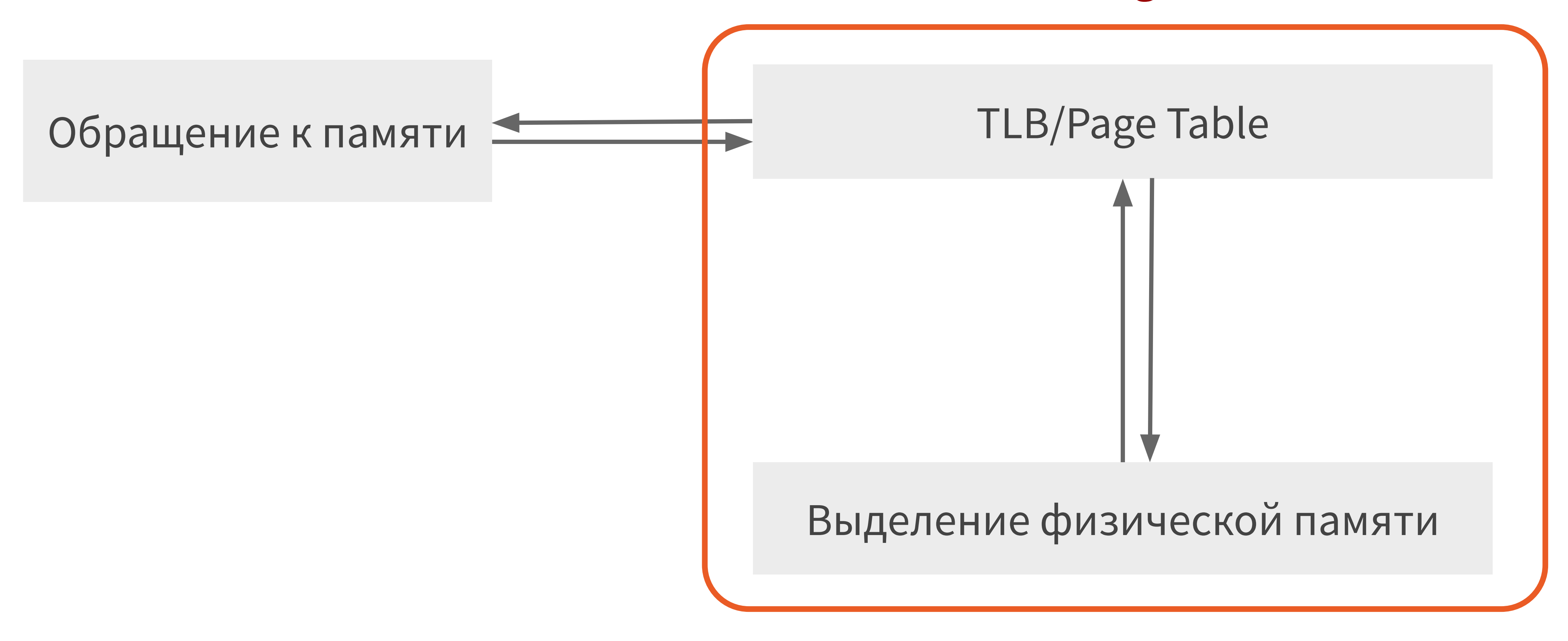
TLB/Page Table

Выделение физической памяти

Page Fault


Трансляция виртуального адреса на физический
Обращение к памяти
TLB/Page Table
Выделение физической памяти

Page Fault


Minor Page Fault
Обращение к памяти
TLB/Page Table
Выделение физической памяти

Non-Temporal Stores + mmap: перфоманс

- горячий page cache

Без учета mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.72	0.72

С учетом mmap/munmap

	mmap+NT, sec	criticalRead, sec
5.5GB	0.82	0.72

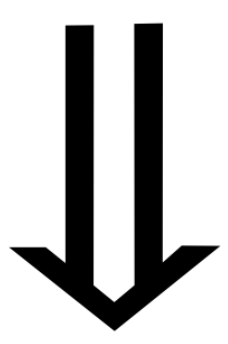
Без учета Page Fault

	mmap+NT, sec	criticalRead, sec
5.5GB	0.51	0.72

Non-Temporal Stores + mmap: резюме

Page Fault

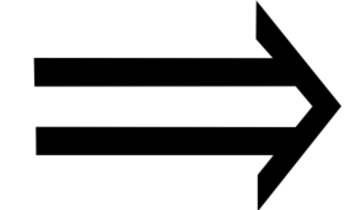
Cache Miss



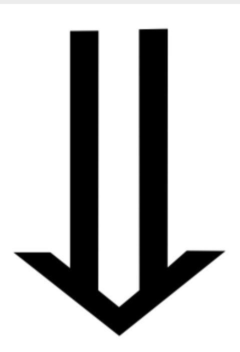
Не подходит для чтения файла

Non-Temporal Stores: резюме

Не кэшируются



Уменьшение загрязнения кэшей



Идеальный выбор для стриминга

Оптимизация производительности: резюме

- Определение изменившихся файлов 
- Определение предыдущей позиции стриминга 
- Чтение новых данных 
- Non-Temporal-стриминг обработанных данных 

Итоговая схема система стриминга

Мониторинг

Локальный диск	inotify	Получение новых данных
	fstat	
	JNI Critical	

Чтение

Обработка

Non-Temporal Store

Буфер на стриминг

Итоги

- Экономия на инфраструктуре до 40%
- Авто-скейлинг
- Уменьшены затраты на стриминг
  - Исключены накладные расходы на I/O
  - Оптимизировано использование кэша

Заключение

Повышение производительности за счет:

- ➔ Выхода за границы абстракций JDK
- ➔ Использования возможностей железа и O/S
- ➔ Обхода особенностей JVM



Q&A



Thank you!

www.griddynamics.com